

evolution@home: Experiences with work units that span more than 7 orders of magnitude in computational complexity

Laurence Loewe

Microbial Ecology Group, Department of Biosciences, Technische Universität München
Weihenstephaner Berg 3, 85354 Freising, Germany
Laurence.Loewe@evolutionary-research.net

Abstract

Individual-based models in evolutionary biology easily lead to multi-parameter applications that need global computing power to exploit their full potential. Mainly due to varying population size parameters, they easily generate computational complexities from less than a second to more than 100 years in case of the Simulator005 of evolution@home. The poorly understood biology of the system leads to automated predictions that may be way off. This report describes first experiences of a global computing system, where users can choose between tasks of different complexity. Besides theoretical complexity limits of tasks that fit global computing, choices of users are analyzed. Potential of incomplete results to increase prediction accuracy is discussed as well as benchmarking computer systems that vary nearly 2 orders of magnitude in their idle processing power. Finally, prediction accuracy is analyzed with the help of a newly defined parameter: error of magnitude. It is concluded, that global computing has great potential for projects with poorly predictable single-run-complexities, if frameworks are designed to allow users to choose their commitment, and if they make use of incomplete results to improve predictions.

1. Introduction

Most evolutionary processes generate a bewildering array of complex phenomena that do not easily lend themselves to mathematical analysis. As further research is often limited by human mathematical capabilities, the advent of powerful computers has generated opportunities for extending the range of problems that can be tackled. Individual-based models (IBMs) have been particularly successful in this respect [1-3]. When properly handled, they are powerful tools for testing evolutionary and ecological hypotheses. However, efficient large-scale work with IBMs requires (i) a framework that facilitates implementation of biological hypotheses in an object-oriented programming language and (ii) enormous

computing power to run enough simulations for a test of the hypotheses implemented. Only emergence of evolutionary bioinformatics may be able to meet these needs [4].

The framework behind evolution@home fills this gap by providing classes and tools that encapsulate biological concepts and facilitate globally distributed computation [4]. The evolution@home system started its first semi-automated global computing project in April 2001 [5]. Although no user-contribution statistics have been published up to date, and no advertisement has been made besides an email to global computing review sites [e.g. 6], more than 150 participants have contributed about 13000 public simulation results with more than 10 years of computing time in 37 weeks. While this underscores the importance of advertisement, press releases, public participant statistics, and a fully automated worker software, it is striking that the largest global computing project SETI@home (3 million PCs, 500000 CPU years in ca. 1.5 years, see <http://www.ud.com>) has a similar ratio of one-time to continuous participants: about 10%.

While other global computing projects have relatively uniform work unit sizes, problem domain inherent constraints make work units of evolution@home span more than 7 orders of magnitude, if a work unit is defined as single complete simulation. This paper reports first experiences of a global computing system with such diverse work units.

2. How to distribute computation of individual-based models

A short review of opportunities for parallelizing evolutionary IBMs helps to see the broader context.

Different models should be run in parallel. As modelling is an art, different researchers will and should build different models of biological reality to gain more confidence in overall predictions. In evolution@home, each simulator implements *one* model that gets a unique SimulatorID from *evolutionary-research* [5], the initiative that runs evolution@home.

Different single-runs should be run in parallel.

Uncertainty in our knowledge requires computation of many different parameter-combinations for each model, and stochastic simulations need to be run several times to estimate distribution of results. A single-run can be viewed as a description of a particular world history that is independent from all others. Thus, each simulator is a so called embarrassingly parallel or multi-parameter application. Parameter combinations are organized into projects. A ProjectID is allocated by *evolutionary-research* to facilitate analysis. Simulations analyzed in this paper belong to project P1 of simulator S005.

Time-series analysis should be distributed. As single runs generate much more potentially interesting data than can be transferred to or handled by central servers, simulators have to do a large part of the analysis themselves. Part of the framework behind *evolution@home* is a set of tools that automates pattern recognition in different time series of parameters that are observed during a simulation. Thus only a small set of pattern-summarizing results is transferred to the central server for further analysis.

Speed-up of single runs is hard. As high bandwidth is pivotal to distribute computation of a single run, currently only SMP nodes may do it. However, as this load-balancing problem is not easily solved in a general way, it will take some time until corresponding simulators will be developed. The same is true for simulations of multi-level population dynamics via hierarchical global computing. Vectorization, finally, is no rewarding concept for stochastic IBMs, as frequent branchings and non-local memory accesses usually destroy speed advantages of vector processors.

Variable work-unit are elegant here. Most global computing projects can divide their work into more or less similar sized work units. Thus, only individual CPU-speeds influence computing time. With long-term evolutionary IBMs the situation is far more complex. To keep all necessary data together, the most natural choice is to run a single simulation on one computer only. To continue extremely long runs on other computers would lead to prohibitive communication and other server side costs, as the longest runs are usually those that generate huge amounts of temporary data. Therefore, users should choose their commitment in terms of physical RAM and computing time to increase the probability that a given run would be completed. To this end RAM and computing time complexity is predicted for each parameter combination. In S005P1 population size RAM requirements range from 1 KB to 900 MB with predicted computing times between 0.01 sec and 180 years. While evolutionary questions would still ask for more, practical considerations momentarily excluded all simulations with more than 30 days predicted computing time. Simulations with less than 15 min were run on a non-public computer. While one can debate on their exact location, lower and upper boundaries indisputably exist for tasks, whose global computation is feasible.

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

$$T_{CPUlow} = \text{Transfersize} / \text{Transferspeed}$$

where *Transfersize* is the sum of all Bytes of all communication needed to distribute the task and collect the result over a network with *Transferspeed* as real-world bandwidth in Bytes/sec. Thus it makes no sense to distribute work units of less than 25 sec expected computing time over a 28K modem line if they generate 100 KB of traffic. While this figure shrinks to 1 msec with Gigabit Ethernet, it rises considerably for semi-automated projects, where occasional manual interaction is needed.

The upper boundary $T_{CPUhigh}$ can be derived by considering Moore's Law: If computing time doubles every T_{Moore} e.g. 2 years, then for extremely long computations, it merits to wait. The break even point is reached at

$$T_{CPUhigh} = 2 \cdot T_{Moore}$$

as a 4 year simulation started today will need only 2 years when started in 2 years on a on a computer that is twice as fast. While, practical considerations usually reduce feasible computing times significantly below this theoretical limit, a global computing framework for evolution needs to take it into consideration.

Incomplete runs can be used. Computers may crash, users can stop a run, or they change their commitments. Any of these reasons can lead to incomplete simulations. While many distributed computing projects cannot use such data, *evolution@home* can. As searching parameter space is one of the most important objectives and computing time predictions are often very crude, incomplete runs can be used to refine predictions, because first glimpses at results usually allow much easier extrapolation, than *ab initio* estimations. Intermediate results have a special parameter value that identifies them. They can be included in many analyses, as they have the same trend indicators as all other results.

3. The Users-Free-Choice System

Currently, *evolution@home* allows users to choose their commitment by downloading one of about 50 run-files from the web with up to about 60 simulation tasks in each. To generate these, parameter combinations of one RAM complexity class were ordered according to predicted computing time and then sequentially packed into one or more run-files. Thus one manual interaction (download run-file and submit results by email) allows comfortable scheduling of many simulations. Simulation

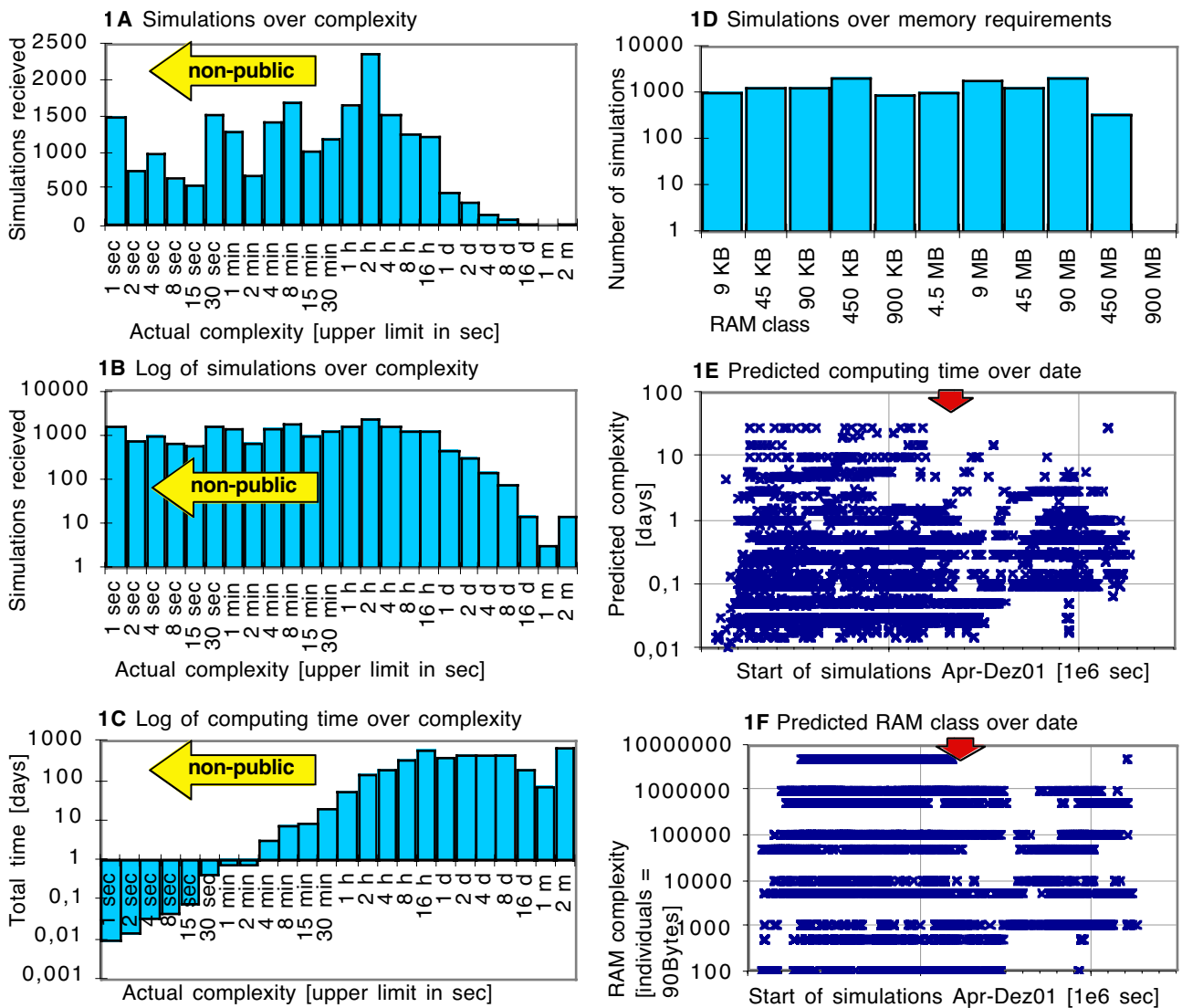


Figure 1: Distribution of user choices. Of about 22000 total simulations, 9000 non public simulations have been excluded from D-F and marked in A-C. The arrow in E-F denotes the first removal of completed run-files from the web (1 x-unit=11.574 days).

start-codes contain complexity predictions for users who want to compose their own special run-files (documentation supplied). While this was probably rarely done, many participants made extensive use of their right to choose. Email feedback confirmed that users like this flexibility, although rather in a fully automated system.

Several issues come up, when users can choose the complexity of their contributions: (i) Biased user choices (ii) Faithfulness of users regarding their commitment (iii) Benchmarking computer systems and (iv) Accuracy of biological foundations of predictions. In the remainder of this paper these issues shall be discussed in light of experiences with evolution@home.

3.1. Biases in user choices

It was initially unclear, whether user bias would be so strong that only a small part of the complexity spectrum would actually be sampled. If one now considers complexity of runs, this is not the case. Figure 1A shows a histogram of the number of simulations with observed computing times that approximately double from one category to the next. The left part of the figure includes the more than 9000 non-public simulations of very small complexity for comparison. These represent a more or less unbiased spectrum of the complexities scheduled. When this variation is taken into account, user choice was more or less uniform for all simulations up to about 16 hours (Figure 1B). Longer computing times were submitted less often with increasing complexity. This is easy to understand, as users picking complex runs will not submit as many single-run-results as users picking small runs. Therefore, commitments appear to be quite uniform when total computing times invested in the corresponding categories are considered (Figure 1C).

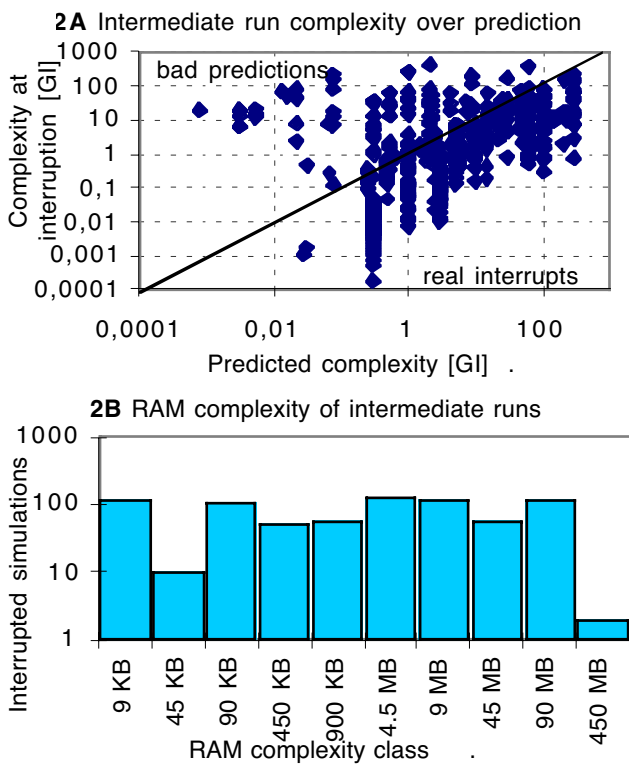


Figure 2: Intermediate Results. **A:** 23% of all interruptions may be attributed to bad predictions; the rest is due to technical causes or user decisions; GI = GigaIndividuals, see section on benchmarking. **B:** RAM complexity of interrupted results shows no trend.

Thus, a simulation scheduling system needs considerable fine-tuning capabilities for complex runs, while shorter tasks should be handled largely automatically.

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 MB class and none in the 900 MB class – a clear reflection of current PC memory sizes (Figure 1D). To find out how this more or less uniform pattern is distributed in time, predicted computing times and RAM complexity were plotted over the date when simulations were started (Figure 1E + 1F). While this confirms overall rough uniformity, changes in the run-files distributed over the web can be tracked in such plots with a delay of about 3 weeks – an indicator of current scheduling flexibility.

3.2. Intermediate results

Nearly 6% of results 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 the time and the

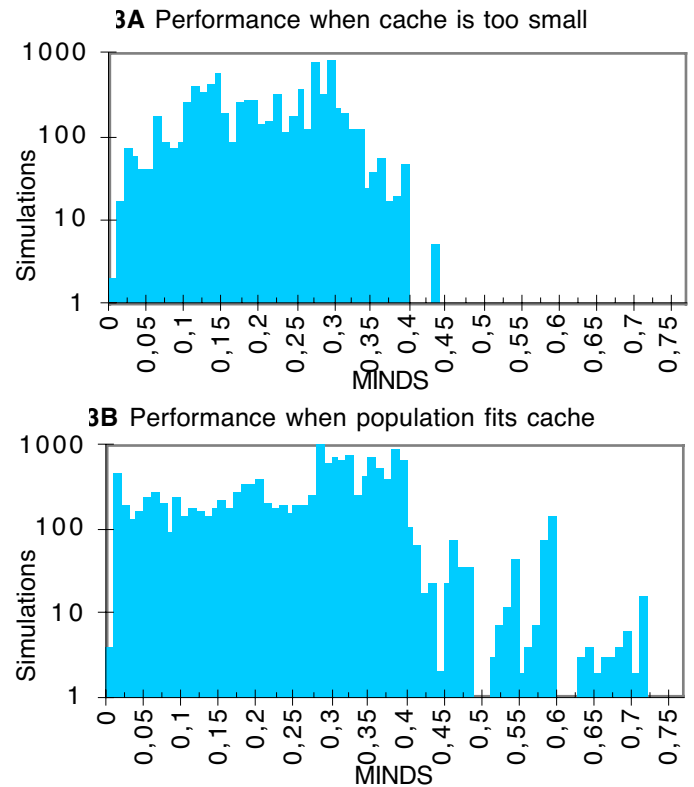


Figure 3: Cache effects on performance distribution. **A:** Little or no cache effects can be expected in simulated populations that need > 4 MB RAM. **B:** Performance of simulations with 1 KB - 1 MB RAM has a high probability to be influenced by cache effects with the fastest systems being ca. 70 times faster than the slowest.

simulator application had not been closed. To avoid losing everything in case of a crash, the list of observed parameters was written to file every hour. Given this fact, it is remarkable that about 94 % of all simulations could run uninterrupted, although many of them had long computing times.

Generation of intermediate results may have two causes. (i) Prediction quality was so bad, that the user stopped the apparently never-ending run, with every moral right on his side. After all, a prediction that fails by 5 orders of magnitude is no prediction at all. (ii) Prediction quality may have been OK, but the user changed his commitment deliberately or involuntary. While every user has the right to do the first, no one can exclude system crashes and other external causes that lead to the second. Figure 2A shows that 23% of all intermediate results may be due to bad predictions, where as 77% come from real interruptions. (Please note that many completed runs had bad predictions too, see Figure 4C.) Figure 2B shows no trend in the memory complexity of intermediate results. At first glance, it would not harm the project, if intermediate results were discarded. In the long term, however, intermediate results

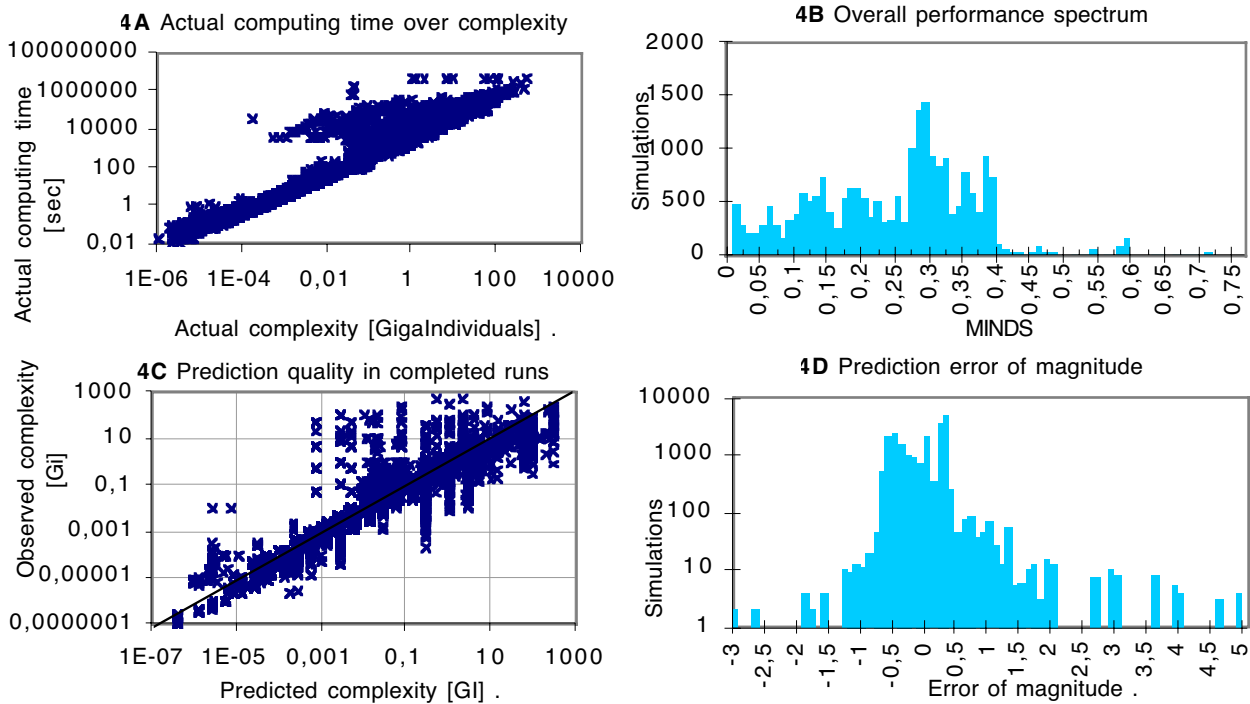


Figure 4: Sources of computing time variability.
A: The nice line in the lower part represents maximal performance of the computer used for small non-public simulations. As the simulator uses the lowest possible background priority, other software may cause considerable slowdown. **B:** Overall performance spectrum. **C:** Actual complexity may considerably deviate from predicted complexity. **D:** Histogram of the error of magnitude (see text for definition) of completed runs only. Although most predictions are accurate within half an order of magnitude, outliers can be extreme.

permit improvement of computing time predictions. This is especially important for models in evolutionary biology, as their behavior is often difficult to predict.

3.3. Benchmarking computer systems

Benchmarking is necessary to give the user a prediction of computing time on *his* system, and importance of hitting at least the correct order of magnitude should not be underestimated. Unfortunately, comparison of computing systems is very complex and many of the benchmarks devised concentrate only on a small part of the performance spectrum [7]. Thus, some have reached the conclusion that the only valid benchmark is the final application. In global computing things get worse, as only idle CPU-cycles are used and execution of any other software decreases performance. Simulators of evolution@home have even more problems. As their tasks span nearly 6 orders of magnitude in RAM complexity, cache effects can play an important role.

Currently the problem is solved as follows. When the simulator is started for the first time, it asks for a

maximal RAM limit on this computer to avoid use of virtual memory, an effective knockout even for fast preemptive multi-tasking operating systems. Before the first simulation is started, a benchmark population is generated that shall (i) test whether allocated memory is really available and (ii) measure 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 an individual prediction whenever the user requests one. 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 most natural basic measure of complexity for individual-based models is one individual whose transition from one moment of simulated time to the next is computed. The number of such “individuals” that can be computed in one second is called INDS or MINDS, if a million individuals are used. When future simulators implement models with many different individuals, INDS refers to a well-defined average standard individual that is characteristic of the corresponding simulator and is used in its complexity prediction formulas. One measure of computational complexity is GigaIndividuals (GI).

Equipped with this background, the performance spectrum of all single-run-results of evolution@home can be understood (Figure 3 and 4AB). Performance of computers that contributed results spans nearly two orders of magnitude, if cache-sized simulations are considered. For simulations that are too large for current caches, performance still differs by a factor of 40. Using Moore’s

Law, this figure can be used to infer that participants of global computing projects use systems that range from brand new to about 10 years age ($40 = 2^{3.3}$; $5.3 \cdot 2a$) – assuming that the slower runs were not due to busy CPUs. Global computing may have to deal with a broader performance spectrum than other applications as people may use it to give their old PCs a new destiny.

Altogether, these results underscore the importance of taking local performance into account when predicting total run time – especially for complex simulations.

3.4. Prediction accuracy

Besides local performance, biological knowledge about the system investigated profoundly influences prediction accuracy. In many simulations that tackle long-term evolutionary questions, models have a general structure that waits for certain events. In most cases, some related parameters indicate arrival of these events. However, to schedule simulations one cannot start them simply to look for the first hints at ultimate complexity (one may need a global computing system just to do this). Thus *very* simple formulas have to be used to arrive at reasonable predictions. However, systems where such simple formulas give the correct answer are usually not 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 many problems in evolutionary biology, especially not for Muller's ratchet, the topic of Simulator005 [4]. Thus, participants and administrators have to live with this inaccuracy. After all, no simulations would have been necessary, if the answer had been known already.

To estimate prediction quality of Simulator005, predicted complexities were compared to actually observed complexities for completed runs. As can be seen in Figure 4C, considerable prediction errors exist in both directions. Unfortunately, one cannot just compute the relative error, as it generates seriously misleading figures due to its inherent asymmetry ($[10^{-7} \text{ observed} - 1 \text{ true}] / 1 \text{ true value}$ leads to 1 as relative error, although the observation is 7 orders of magnitude off and would lead to a relative error of 10^7 , if 10^7 had been observed). However, as all relevant values have the same sign, the **error of magnitude**, Err_{mag} , can be defined as

$$Err_{mag} = \text{Log}_{10}(1 + |C_{observed} - C_{true}| / C_{true})$$

where C denotes the observed or predicted (= 'true') computational complexity in GigaIndividuals. Err_{mag} has the intuitive feature of being symmetric around 0, where 0 indicates no error, negative or positive values indicate that the observation is below or above the prediction, respectively. Furthermore, an Err_{mag} of 1 or 3 has the intuitive meaning of missing the mark by a factor of 10 or 1000, respectively. Figure 4D shows observed errors of magnitude. Most predictions deviate by Err_{mag} 0.5 or

less. This accuracy within a factor of 3 is remarkable, given the fact that we are dealing with more than 7 orders of magnitude here. However, currently observed extremes range up to 3 orders of magnitude below and 5 orders of magnitude above the predicted value.

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

4. Related work

Although there are too many global computing projects to discuss them here, a few aspects are worth mentioning (see [6] for a review and for links to the following projects). Computational complexity of individual work units from global computing projects 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 of the climateprediction.com project). 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 computation of their large numbers of work units. Thus, they share the application centric view of Application Level Scheduling (ALS), an important topic in computational grid research [8, 9]. In contrast to classical resource or job scheduling on supercomputers, ALS optimizes timely execution from the perspective of the application by e.g. avoiding delays due to batch queues or slow connectivity. Optimization criteria may 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 like CPU and Network load. These observations are then used to compute the best schedule under given circumstances. For an example, see "AppLeS" [9]. Such an ALS system may allow users to work (more or less) interactively with applications that are too complex for any single workstation (e.g. [10]). Recent efforts to support parameter-sweep applications [11] extend ALS to extremely complex problems that are similar to most global computing projects.

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. These problems require fast processing of the most important work units in the first place, while overall completion time is less important. As interesting work units may speedup overall progress of the project or point to interesting regions of parameter space for more intense investigation, they play a key role in steering the whole project. However, 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. This has been the case with *evolution@home*. Once these data handling issues are solved, more sophisticated approaches for scheduling can be envisioned.

In the mean time, the User-Free-Choice System presented in this paper has a distinctive feature: simplicity. It neither needs a specially adapted Scheduler, nor a Network Weather Service, nor a list of known computers, nor standing Internet connections. Thus it can be operated with a simple static web server on the operators side and occasional dial-in connections on the participants side. Furthermore, it allows participants to remain anonymous, should they so wish. Thus administrative overheads for scheduling are minimal, as are requirements for participants. If turnaround time is less critical than a low implementation complexity and easy access to more computing power, then the minimalist approach of the User-Free-Choice system can be recommended. In other cases more sophisticated ALS will be necessary. Neither, however, helps to organize the flood of data produced.

5. Lessons and Conclusions

It was not clear initially, whether users would devote computing time to all complexity classes or rather stick with the most simple tasks. The bandwidth of received results was encouraging and suggests that stochastic parameter-sweep applications may use a similar system, if they need to repeat their runs many times anyway, and if there is no pressing deadline for getting particular results.

A weakness of the current system is its dependency on manual analysis of results. This leads to considerable slowness in updating progress and run-files on the web. Ideally, the website would contain only the hottest run-files of the hour making sure that every simulation with a particular complexity is computed once, before any is computed twice. Such a scheduling mechanism, however, needs to be tightly integrated with the whole *evolution@home* results analysis workflow and cannot be easily imported from outside. Thus the next steps for

evolution@home include implementation of such a system.

Another important feature, however, 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.

All in all, *evolution@home* is the first global computing system that has to deal with work units that span more than 7 orders of magnitude of computational complexity. Initial experiences reported here, indicate that global computing has great potential to develop the flexibility needed to cope with such diversity, if frameworks incorporate the corresponding features.

Acknowledgements

A big thank-you to all participants of *evolution@home* for contributing more than 10 years of computing time. Special thanks also to S. Scherer and the FML Institute of Microbiology at TUM for financial support, and to three anonymous referees, Franck Cappello and Nigel Crompton for their comments on an earlier version of the manuscript.

References

- [1] O. P. Judson, "The rise of the individual-based model in ecology," *Trends Ecol. Evol.*, vol. 9, pp. 9-14, 1994.
- [2] J. McGlade, "Advanced Ecological Theory." Oxford: Blackwell Science, 1999.
- [3] D. L. DeAngelis and L. J. Gross, "Individual-based models and approaches in ecology." New York: Chapman & Hall, 1992, pp. 525.
- [4] L. Loewe, "Evolutionary bioinformatics: Predicting genetic stability of asexual genomes by global computing," PhD Thesis, *Wissenschaftszentrum Weihenstephan*, Freising: Technische Universität München, 2002, in preparation.
- [5] L. Loewe, "The *evolution@home* website," <http://www.evolutionary-research.net>, 2002.
- [6] K. Pearson, "Internet-based Distributed Computing Projects." <http://www.aspenleaf.com/distributed/>, 2002.
- [7] J. L. Gustafson and R. Todi, "Conventional benchmarks as a sample of the performance spectrum," Hawaii International Conference on System Sciences, <http://www.scl.ameslab.gov/Publications/HICSS98/HICSS98.pdf>, 1998.
- [8] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, "Application-Level Scheduling on distributed heterogeneous networks (Technical Paper)," Proceedings of Supercomputing '96, 1996.
- [9] F. Berman, "Application Level Scheduling on the Computational Grid," <http://apples.ucsd.edu>, 2002.
- [10] S. Smallen, W. Cirne, J. Frey, F. Berman, R. Wolski, M.-H. Su, C. Kesselman, S. Young, and M. Ellisman, "Combining Workstations and Supercomputers to Support Grid Applications: The Parallel Tomography Experience," Proc. 9th Heterogenous Computing Workshop, 2000.
- [11] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, "Heuristics for scheduling parameter sweep applications in Grid environments," Proc. 9th Heterogenous Computing Workshop, 2000.