Virtual Prairie: going green with volunteer computing

Malek Smaoui
University of Houston
Computer Science Department
Houston, Texas 77004
msmaoui@cs.uh.edu

Marc Garbey
University of Houston
Computer Science Department
Houston, Texas 77004
garbey@cs.uh.edu

Cendrine Mony
UMR CNRS 6553 ECOBIO
University of Rennes1
Rennes, France
cendrine.mony@univ-rennes1.fr

Abstract

Multi-species prairie composed primarily of clonal plants can achieve different ecological functions. They can be a good source of biofuel, a mean of water purification in agriculture or a biodiversity guardian. The goal of the Virtual Prairie project is to study the fundamental mechanisms involved in the population dynamic of a prairie and the optimum engineering design of prairies serving these and other functions. Our study is a modelization and a simulation of the growth of such clonal plants in isolation or competing with each other. This study is a very compute intensive enterprise. However, it can be eventually managed as an embarrassingly parallel set of jobs. Therefore, we adapted our simulation project to run on BOINC, a platform for volunteer computing, which compute nodes are reached using the commercial Internet connections. To our knowledge, this is the first ecology project that uses volunteer computing and it is the largest simulation ever done on plant ecology. Our preliminary results show that there is no unique optimum design solution but rather different optimum solutions that can co-exist.

1. Introduction and motivation

New ecological questions emerge due to the increasing social demand for solutions to environmental problems. National governments are more and more concerned by the urgency of contributing to a sustainable development at the planet level in order to compensate pollution and global warming. In this framework, research to reduce pollution has to focus on finding low cost solutions to priority issues in a very short term. Ecological problems have the specificity of being dependent on biological elements with complex interactions and which response may be delayed at a year or multi-year scales due to biological cycles. New tools are therefore needed to go past over biological constraints and take into account the complexity of living ecosystems. Modelling may achieve these challenges and enable to

(i) provide virtual experiments for extensively testing solutions in a shorter time than what would have been possible through real experimentation;

(ii) optimize experimental designs to obtain the maximum results with the minimum number of experiments.

We are developing a computational modelling of terrestrial plant community ecology via the simulation of a prairie (e.g. interactive terrestrial plant populations) in relation with environmental data.

In this paper, we present a volunteer computing solution for the modelling of this ecological problem. The project, called Virtual Prairie (ViP), aims at studying the fundamental mechanisms involved in the development of prairies and consequently designing prairies serving different ecological functions.

• According to a recent study [19] sponsored by NSF, “Mixed Prairie Grasses are better source of bio-fuel than corn ethanol and soybean biodiesel”. Tilman et al. advocate that optimized mixed prairies are carbon-negative and provide more usable energy per acre than other bio-fuel sources based on cereals.

• Strips of prairies at the border of crop fields can improve water purification from nitrate in excess before it
reaches the streams [6, 7]. The recent EU agrienvironmental policy encourages farmers to plant these prairies, but the optimum mixed prairies that minimizes the land used to that system is yet to come.

- Biodiversity is one of nowadays concerns. Grasslands are an habitat for many plant and animal species providing an adapted management. Designing plant cover that provides this habitat and conserve vulnerable species is of key interest [9, 12, 18].

The attempts to model prairies [16, 17, 5] have been limited so far to small scale simulation. This may be due to the unavailability of important computing resources necessary for simulating complex prairie models. A typical model in plant ecology involves many parameters and it is clear that one can not rely on simulations that explore a small part of the parameter space. In the mean time browsing the parameter space and/or using a genetic algorithm for optimization can generate a very large number of tasks with embarrassing parallelism.

We found that volunteer computing could be a good solution for this issue and we adopted BOINC as our computation platform. Actually, the general public environmental awareness is increasing. People are getting more concerned about environmental issues like global warming. Then, it is easy to get them involved in such a project with some advertisement and information campaigns. We can further reinforce the interest of the public to environmental issues with such initiative. We are going now to discuss our model for a virtual prairie.

2. Prairie modelization

Most herbaceous species in many plant communities are able to reproduce through vegetative iterations of ramets connected through aboveground or underground organs (c.f. Fig. 1). These are called clonal plants. Each individual is a network of ramets and connections through which resources and information can be shared or stored and which enables space colonization. Prairies are constituted mainly of clonal plants and; understanding clonal strategies in such ecosystems is a first goal of this project. The study needs two main steps:

- Simulating the growth of individual isolated clonal plants and analyzing their behavior according to the species characteristics and the environment and stress factors.
- Simulating the development of mono and multi-species prairies involving many individuals competing for the same resources and space.

The growth of individual clonal plants follows a classical approach described thereafter.

2.1 Individual clonal plants simulation

For this first step, we implemented an individual based model (IBM) [5, 8, 10, 11] which is a set of parameterized rules representing the metabolism process (carbohydrate production, ramet and connection costs of production), the plant architecture (ramification, elongation processes) and its resource strategy (resource sharing and storage).

The plant is represented by a set of cells or modules. Each module is either a ramet module or a connection module (c.f. Fig. 2). However, keeping the continuity on the placement/locations of these modules gets us into losing the control over the model and makes it harder to collect the relevant information about the plant. That is why the model is transposed on a grid (discretization). We select an hexagonal cell set up that has 6 privilege directions. Indeed, the plant grows inside a square with a hexagonal grid (100x100) and its modules are placed on the nodes of the grid (c.f. Fig. 4). These modules are added gradually during the simulation. The size of the grid is large enough to capture the behavior for the time scale under consideration i.e one season.

At $t_0$, the plant has only one initial module (considered also as a ramet module). Plant spatial colonization is di-
Figure 3. Experiment companion by the image segmentation of the clone

rectly linked with branching and elongating patterns. The creation of a new cell is made either by elongating an existing connexion or by branching at an existing ramet node in order to initiate another connexion. Thus, at each time step, the IBM decides either to add a connection or a ramet module or no module at all, depending on the relative cost of production of this new cell and the resources available for it. The choice of a ramet vs. connexion cells depend on the inter-ramet distance set up at the beginning of the simulation. The location of this new cell via elongation or branching depends on several factors:

- The order of the branch (primary, secondary, tertiary).
- The length of the branch.

For elongation process, the location of the new module follows the direction of the branch. For the ramification process, the location is a random choice between 2 directions. The simulation is done on 100 time steps which corresponds to one season. At a given time \( t \) and before adding a new module, all the ramet modules found on the grid accumulate biomass by photosynthesis.

All the rules of the model obey a parameterized probability distribution. This set of rules is based on the literature, but also on a set of controlled experiments that we have done. Fig. 3 shows the experimental set up.

Experiments with various species and environment were run and the structure of the clone was extracted assisted by image analysis [4]. This type of experiments is however very tedious and time consuming, and we cannot afford to have as many individual plants as we would prefer. However our modelling is guided by this set of experiments, can be used to test scientific hypothesis qualitatively and/or do reverse engineering to match our observation. Conversely, this set up allows us also to prepare the next campaign of experiments by narrowing our attention on what the simulation suggests to be the dominant phenomena.

Fig. 4 is an example of the graphic output of a simulation. We are going now to present the virtual prairie that integrates this individual based simulation.

2.2 Multi-species prairie

For the second step, we supplement the previous IBM with rules representing the interaction between plants and the competition between species. In fact, we consider this time a significantly larger surface where we distribute randomly a given number of initial ramets (hundreds to thousands). Each initial ramet is encapsulated inside a square as previously. At each prairie-scale time step, we grow the plants one by one using the plant-scale time steps. The order followed for growing the plants is determined randomly at every prairie-scale time step. Additional IBM rules exclude for example the superposition of ramets of different plants. Fig. 5 is an example of the graphic output of the application for a prairie with 100 individuals. More details on this application will be published in a companion paper [15], but let us now discuss the implications in term of computation cost with the deployment of our numerical experiments.
3 Computation needs for Virtual Prairie

As previously mentioned, the IBM obeys a set of parameterized rules. The dimension of the parameter space for one individual plant can reach 15 and even more. This dimension will depend on the focus of the ecology study. We need to perform two tasks:

- **Task 1** Get a parameter space response i.e. browse the parameter space with selected values for each parameter and get the simulation results for these parameter combinations [14]. This coarse parameter space investigation is completed by a detail analysis that follows.
- **Task 2** Perform an optimization process using for example a genetic algorithm [13] or a particle swarm algorithm.

Both steps are necessary in that order, because the end user in ecology does not know necessarily in advance what emergent properties he is looking for.

Added to that, the IBM is by nature a stochastic algorithm. It relies on random number generation/selection and one simulation of the plant growth is far from being significant. We need to repeat this simulation many times and calculate the mean and the standard deviation of the desired outputs (total biomass accumulated, number of ramet's module, number of connection's module, length of stolon ...) obtained at each simulation. Actually, part of testing the application was verifying that the mean and standard deviation on the output values tend to a fixed value as the number of stochastic simulations increases.

Because this convergence that was monitored is slow, like for Monte Carlo methods, we decided to simulate the growth of a plant with 1000 runs for each parameter setting. Nevertheless, a 100x100 grid with 100 plant-scale time steps that represents a little bit less than a season is still very fast. On an Intel Pentium 1.7GHz with 1GB of RAM and a Windows XP platform it takes on an average 45 seconds. Although this metric depends enormously on the combination of parameter values input of the model, it proved to be a good estimation when we run the computations on the volunteered computers using BOINC. Browsing a parameter space of dimension 15 with only 3 values per parameter takes about 14 million simulations. These simulations will take around 20 years on the previously mentioned single computer. With a 72-nodes dedicated cluster (of the same type of the previously mentioned computer), it will take more than 3 months.

A simulation of the growth of a mono-species prairie of 100 plants with 1000 (stochastic) runs in a 600x700 grid and during 100 prairie-scale time steps takes significantly more time. This is due to two main reasons:

- The actual design of the application induces significantly more memory accesses to manage the competition for resources between individual plants.
- With the increased size of the global problem, it does not fit into the cache anymore. Then a lot of swapping between the cache and the main memory occurs. This part might be optimized more or less depending on the density of plants.

The application design needs a little tweaking in order to reduce significantly this computation cost which is several order of magnitude the elapse time of our first step with isolated plants.

Let us restrict ourselves to Task 1. Overall, the corresponding huge number of simulations are all independent from each other. This means that we have a huge number of embarrassingly parallel computation jobs. So, volunteer computing fits perfectly this situation and with the computing and storage potential of this resource, we can expect to achieve the computations in a reasonable time. We are going now to describe that technique.

4 The Virtual Prairie BOINC project

BOINC stands for the Berkeley Open Infrastructure for Network Computing [2, 3]. It is a middleware that enables the distribution of embarrassingly parallel jobs on volunteered computing devices over the Internet.

4.1 Volunteer computing with BOINC

Volunteer computing is an arrangement in which people (volunteers) provide computing resources to projects, which use the resources to do distributed computing and/or storage.

- Volunteers are typically members of the general public who own Internet-connected PCs. Organizations such as schools and businesses may also volunteer the use of their computers.
- Projects are typically academic (university-based) and do scientific research. Several aspects of the project-volunteer relationship are worth noting.
- Volunteers are effectively anonymous; although they may be required to register and supply email address or other information, there is no way for a project to link them to a real-world identity.
- Because of their anonymity, volunteers are not accountable to projects.

Some of the early projects which used volunteer computing include distributed.net, SETI@home, and Folding@home [1]. They are behind the idea of the development of BOINC as a middleware for volunteer computing developed by the team of Dr. David Anderson at the Berkeley Space Science Lab. Nowadays, there are at least 50 volunteer computing projects in different research fields. These are all independently operated, see http://boinc.berkeley.edu/.
BOINC has a server/client architecture (c.f. Fig. 6). The volunteers install the client on their computers and decide to volunteer their resources to the projects of their choice. The client will then download jobs from the different projects, compute them and report the results back.

The server side hosts a web interface, a data base and a set of functionalities for the distribution of jobs and aggregation of results. The web interface allows the users to get information about the science involved in the project, consult the work status of the project, communicate with each other and with the project administrators via a message board, consult and edit their accounts and profiles, etc.

To handle the work distribution and aggregation, BOINC uses abstractions which are mainly “workunits” and “results”. A workunit is a reference to an application and an input file. A result is a reference to a workunit and an output file. Since the resources (volunteered computers) are not necessarily reliable, BOINC uses redundant calculation to ensure the correctness of the results obtained for each workunit. So, for each workunit, a predefined number of results are sent to different users and then their output files are compared and validated if they agree.

A set of daemons running on the server interacting with a data base work together to ensure all these mechanisms. A feeder is responsible for creating the needed results for each workunit according to its status. A scheduler is responsible for dispatching these jobs (results) to the clients [3]. A “validator” is responsible for comparing the results for every workunit and validating or invalidating them. An “assimilator” is responsible for the post-processing of the validated results on the fly. These demons “communicate” with each other by updating the status of the workunits and the results in the database.

We are going to report now on our experience with BOINC.

4.2 Volunteer computing for Virtual Prairie

Although BOINC is a good solution for many scientific problems, installing the server components is not an easy work. In fact, BOINC depends on many other open source softwares that have to be on the host system like for example apache as a web server and mysql as the database management system. Also, in order to create a project and get the sample application to run, it is necessary to read somehow in detail the BOINC documentation. But this investment largely pays off, because this system works and delivers routinely teraflops!

4.2.1 Virtual prairie implementation for BOINC

Starting with the first step of the project, the virtual prairie application simulates the growth of an isolated individual plant in a 100x100 square grid. The BOINC-adapted Virtual Prairie application reads from the input file’s 15 simulation parameters. It calculates the mean and the standard deviation of 12 characteristics of the plant. The output file consists in a matrix row starting with the list of the input parameters then the 24 (12 means and 12 standard deviations) output values. These matrix lines are grouped into one matrix in order to perform a statistical study and the ecology analysis. We are going to describe the result of our first campaign of numerical experiments.

4.2.2 First computation campaigns

For the very first computation campaign, we generated about 5.5 million parameter combinations using 2 to 4 values per parameter. In order to reduce the network traffic, we grouped them into jobs of 25 simulation runs with 20 different parameter simulations. We had application versions for Windows 32 bits and Linux 32 bits platforms. We used some ready-to-use programs that come with BOINC for the validation and assimilation of the results. At first we did not use any redundancy because we could not figure out a way to initialize the seed of random number generation on the wingmen (computers computing results for the same workunit) with the same value, without biasing the whole lot of jobs. So, the validator was validating all the results produced by the users and the assimilator copying the output files obtained to a specified location. Consequently, we had difficulties in gathering the whole result matrix 5.5 million lines. First, we had to write separate scripts to concatenate the lines of all the output file into just one big file. Then, we had to verify any obvious errors in the final file and rerun those missing results on our local network of computers. An analysis of these preliminary results did show that we could have chosen better our parameter set and that some combination were giving useless information. It could have been thought upfront, but the IBM is still a rather complex algorithm.

For the second computation campaign, we generated more than 18 million parameter combinations. We added application versions for 64 bits platforms and this time we enabled redundant computation. Indeed, the seed for the random number generation was chosen for each workunit at its creation process, stored as an attribute (of the workunit), sent with the job to the client then passed as an argument.
to the application when the computation starts. The validator compares output file from obtained results byte-by-byte, so we used a feature called homogeneous redundancy. This feature tells the feeder and the scheduler to create and send results for the same workunit only to similar platforms. Actually, results produced by two different platforms may be a little different and can be still both correct. This is what is called numerical discrepancies. So, sending jobs/results for the same workunit to only similar platforms avoids invalidation of correct but different results due to numerical discrepancies. We also added our own assimilator which concatenates the validated output files on the fly. All this avoided us the error verification and the manual concatenation when the computation is done.

The first computation campaign took 1 month to complete with 70 local clients (computer science department lab at University of Houston) during the first 20 days and an average of 600 worldwide clients for the last 10 days. We started to get more and more clients afterward. So, the second campaign lasted 45 days on an average of 2000 clients. We are going next to describe what we consider still to be a preliminary result.

4.2.3 Result analysis

Our first analysis is devoted to the understanding of the plant’s portrait that gains the most biomass. We first reduce drastically the data set to work with by keeping those few per cent simulation results that produces the most biomass. We keep for example the top 0.3 % or our 18 millions simulations which results in 54 thousands parameter combination results! On that reduced data set we can plot histogram of parameter frequencies, and eventually apply clustering techniques.

As for each parameter, we create a histogram of the distribution of plants in the selected set on these parameter values (c.f. Fig. 7). We can categorize the result as follows:

- The selected plants are distributed evenly between the values of the parameter. We conclude that this parameter has no influence on the set of optimum results.
- All the selected plants use the same (centered around the same) value of this parameter. We conclude that we should look for a unique optimum value.
- The selected plants are unevenly distributed between the values of the parameters. It is then possible that several clusters of optimum solutions co-exist.

This last configuration is indeed most interesting to understand the complexity of plant strategy to best fit the environment.

The application of a standard clustering algorithm on our reduced data set of 54 thousands results is visualized in Fig. 8 and Fig. 9.

It shows that we have at least two competing strategies and possibly three but not four. More statistical analysis should be done indeed, and there is no guaranty that the same result will be similar for a virtual prairie with many individuals that have the same characteristics. The individual plant compete to itself in the virtual prairie “population” setting to access a limited resource. Our third campaign of simulation is about to address this problem. Most likely we can have two different clusters of plants that cooperate and get more biomass combined together than separately. This kind of finding do need an enormous quantity of computations and somehow the monitoring of the ecology’s specialist to guide the discovery process. We are going to describe our current step to equip our BOINC simulation with a monitoring tool.

4.2.4 Ongoing work

The above result analysis is all done using MATLAB. The MATLAB code takes as input of the order of a dozen of gigabytes text file produced with one month of BOINC computations. We find this procedure inconvenient for three main reasons:

- We have to wait all the computation period in order to be able to consider the whole results.
- The file obtained from the computation which is at the
same time the MATLAB input is too big to be handled by MATLAB in a one time call. We have to split the file again into chunks.

- Although MATLAB is a very good and easy-to-use tool for testing and prototyping, its performance is so low that it can not be a production tool.

Consequently, we are developing a customized a posteriori analysis tool that is portable and performing. In our scenario this set of statistical tools must be called in a transparent way by a graphic user interface dedicated to the ecologist while the results are coming up. Further the user is informed on the overall progression of the numerical experiment.

A by product of that graphic user interface will be a better demonstration to our BOINC users of our scientific discovery process. Volunteer computing power cannot be bought; it must be earned. A research project that has limited funding but large public appeal can get that way a huge computing power. It is indeed what we need to simulate virtual prairies with thousands of individuals.

5 Conclusion

This paper has presented, to the best of our knowledge, the first large scale ecology project that uses volunteer computing. The considerable high computing requirements of our application fits the features offered by BOINC and makes this platform a viable and economic solution. However, this system needs many project-specific customizations and add-ons to be productive. The deployment of this volunteer computing solution has already given interesting preliminary results on clonal plants. These results need to be consolidated by an optimization process using for example the genetic algorithm and by new experimental data.

References
