Parallel Simulation of the Propagation of Powdery Mildew in a Vineyard

Agnès Calonnec[1], Guillaume Latu[2], Jean-Marc Naulin[1], Jean Roman[3], and Gaël Tessier[3]

[1] INRA Bordeaux, UMR INRA-ENITA, Santé Végétale
BP81, 33883 Villenave d'Ornon Cedex, France
[2] LSIIT UMR 7005, Université Strasbourg 1
67412 Illkirch Cedex, France
[3] INRIA Futurs and LaBRI UMR 5800, ScAlApplix project
Université Bordeaux 1 and ENSEIRB, 33405 Talence Cedex, France
http://www.labri.fr/scalapplix

Abstract. This paper describes a parallel simulator for the propagation of a parasite in a vineyard. The model considers the structure, the growth and the susceptibility of the plant which play a major role in the development of the fungus and the spread of epidemic. Two spatial scales are distinguished for the dispersal of the parasite. We use both a realistic discrete model for the local dispersal, and a stochastic model for the long-range dispersal that averages the displacement of spores. An algorithmic description of the parallel simulator is given and real life numerical experiments on IBM SP3 are provided, that use up to 128 processors.

1 Introduction

In this paper, we consider the simulation of a biological host-parasite system. The studied parasite is *powdery mildew*, a fungus of grapevine. Many epidemiological studies have been performed on this topic; however the dynamics of the spread of epidemics is not well known and powdery mildew is still the main fungus disease of grapevine in the world.

A large number of multiscale mechanisms interact in this system. A better understanding and a more effective control of epidemics will depend on our understanding of the dynamical relationships between the environment, the host and the pathogen. Knowledge obtained during experiments in vineyards will be first integrated into a model and then into a simulator. One purpose of this work is to reproduce the interactive events of the system and to synthesize them in order to understand and evaluate macroscopic emerging phenomena.

The simulation requires a large amount of computations, mainly due to the number of spores produced by the parasite and dispersed over the vineyard. An initial sequential simulator only considered only one grapevine [4]. A parallel version has been developed to model the dynamics of epidemic over a parcel. To our knowledge, this approach based on realistic simulations is rather new and has not been yet met in other research works concerning this topic.

J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, LNCS 3648, pp. 1254–1264, 2005.

The biological system and its modeling will be first presented. A profiling of the sequential program will be detailled so as to identify the most time-consuming steps that will be good candidates for parallelization. Then, data distribution and parallel algorithms will be explained; results about scalability, load-balancing and performance issues of this first implementation will be given. Finally, we will conclude with the possible evolutions.

This interdisciplinary work is a collaboration between the INRIA Futurs ScAlApplix project and the LSIIT UMR 7005 for the computer science field, the INRA UMR Santé Végétale in Villenave d'Ornon for the biological investigations and the MAB UMR 5466 for the mathematical models.

2 Biological Issues and Modeling

The structure of a grapevine is strongly influenced by management practices of viticulture system. Two almost horizontal branches carry primary shoots which themselves carry secondary shoots. Apparition and growth of organs are essentially dependant on their position, on the vigour of the plant and on the temperature.

Powdery mildew [1] is a polycyclic fungus that spreads thanks to microscopic airborne spores. We break up the biological cycle in several processes: infection of leaves or clusters by spores, a latency period during which the rising colony is only growing, and a sporulation phase during which spores are released by wind.

2.1 General Modeling

The simulation covers a single season from January to the beginning of September with a time step of one day. Location and onset of primary infection are parameters of the simulation. The dynamics of epidemic is closely related to the quantitative and qualitative development of hosts: the number, the position and the age of organs. Thus the model simulates the 3D development of stocks. The computer model for a grapevine is a binary tree, in which each node represents an element of the plant. A node contains information on its spatial configuration, its biological attributes, and its possible infection state. Parameterized functions, some of them stochastic, are used to describe system growth. Host growth depends on a few magnitudes: temperature T and trophic state which is a temperature-dependant variable. Fungal colony growth depends on temperature and organ age. A vigorous grapevine can bear hardly thousand leaves whereas a weak one three to four times less. The model restricts infection to leaves and the number of colonies to one per leaf. During a day, approximatively tens of thousands of spores are possibly extracted from all the sporulating colonies of a grapevine.

As for the dispersal of spores, it is for the moment impossible for us to know the real movement of microscopic spores that can travel up to several hundreds of meters. Therefore, two scales for the dispersal of spores have been distinguished: local and long-range dispersals. The limit between these scales remains

confusing as it depends on the studied pathosystem. Literature mentions the ratio 80 %/20 % between local and long-range dispersals for an optimal disease spread [9].

2.2 Local Dispersal

In the current version of the model, local dispersal was limited to the source grapevine and its two direct neighbours in the row. At this scale, the distance covered by spores is short, the dispersal occurs in the canopy, supposed to be homogeneous in a local area. Thus, we made the hypothesis that spores have linear trajectories during local dispersal.

Each day, spores are spreading from each sporulating colony. The spread is performed within a *dispersal cone*. Its axis orientation is determined by the mean wind direction of the day represented by the vector (ux, uy, uz). Its opening angle *alpha* is a simulation parameter. Algorithms and data structures have been inspired by ray-tracing methods in image synthesis [8]. A rectangular parallelepiped delimits the volume of grapevine. For efficiency, this volume was cut out with a discrete mesh size of small parallelepipeds called *voxels* [6]. Each voxel has the list of leaves contained in its volume. So computing the leaves intercepting a cone comes down to getting the voxels intercepting this cone, as shown in Fig. 1. The voxel discretization avoids to traverse the whole binary tree for the determination of all the leaves of the stock intercepting the cone. When a cone reaches one edge of the including parallelepiped, its becoming depends on the exit side: spores either fall on the ground, or they are transmitted to the contiguous grapevine, or they are dispersed over the vineyard.

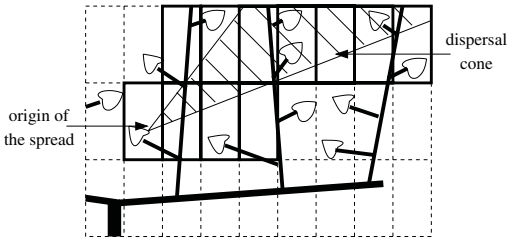

Fig. 1. Dispersal cone in a grapevine with bold voxels intercepting the cone

2.3 Long-Range Dispersal

Field data come from several campaigns of disease follow-up in vineyards. Measures of vertical and horizontal gradients of spore densities over short periods and in fixed vegetation are available, but no spore dispersal measures on a growing crop. At field scale, the previous cone-based dispersal approach is not realistic enough, because the spores have complex trajectories. A stochastic and averaged approach using distribution laws [7] has been considered.

From each grapevine, random drawings following Gaussian distributions yield a displacement; this displacement identifies the destination grapevine for each spore that has to be dispersed over the field. Then, on the destination plant, dispersal cones are used to determine the points of impact of spores, and so the infected leaves.

Set hypotheses and choices of distributions require calibrating the simulator outputs with field data. Some parameters have already been estimated, some others will be refined in the future.

3 Profiling of the Sequential Program

This section is about the time complexity and performance analysis of the sequential simulator. This simulator treats a single grapevine. Of course, there is no dispersal with neighbouring vine stocks or over the field.

During each iteration (one day) of the simulation, variables T, ux, uy, uz are fetched. Before bud break, we evaluate if there is budding or not. After budding, an iteration consists of several steps. Primary infection and management practices occur only at precise day and so, count for a little part of computation time. On the other hand, growth and local dispersal are the most costly steps, that is why they will be examined in detail. To formulate accurately their time complexities, let us call $\#nodes$ the number of nodes in the binary tree, $\#leaves$ its number of leaves and $\#voxels$ the number of voxels in the grapevine volume.

3.1 Apparition and Growth of Organs

Apparition and growth of organs are two sub-steps, each one using one recursive traversal of the binary tree. Host growth and pathogen growth are calculated simultaneously while treating infected nodes. The time complexity of the growth function is $\Theta(\#nodes)$.

3.2 Dispersal of Spores into Grapevine

Input parameters of local dispersal are a grapevine, the day, and the temperature and wind characteristics of the day. Dispersal process is based on a recursive traversal of the binary tree. For each infected node encountered, operations of the function `source_dispersal` (described in Fig. 2) are performed. Variables x, y and z are the node coordinates, and *n_spores* is the number of spores extracted from its colony on the current day. At function return, *n_spores* represents the number of spores not captured and is daily accumulated in the *vine* data structure (see Fig. 3).

The procedure `get_voxels_cone` gets the voxels intercepting the cone issued from the colony in (x, y, z), with the direction (ux, uy, uz) and the opening angle *alpha*. Its time complexity is linear with the number of voxels returned in *vox_list*. This number depends on the position of the source lesion, on the direction and opening angle of the dispersal cone and on the mesh granularity. With our set of parameters, the mesh size is 1.5 m long, 1 m large and 1.4 m high,

```
function source_dispersal(day, T, x, y, z, ux, uy, uz, n_spores)
  vox_list <- get_voxels_cone(x, y, z, ux, uy, uz, alpha)
  nodedist_list <- get_nodedist_list(vox_list, x, y, z)
  quicksort(node_list)
  for nd in nodedist_list do
    if test_node_in_cone(nd, x, y, z, ux, uy, uz, alpha)
      captured <- captured_spores(nd, x, y, z, alpha)
      n_spores <- n_spores - captured
      potential <- node_potential(nd, day, T, captured)
      if(potential >= thresthold) then
        node_infection(nd, day)
    endif
  end for
end function
```

Fig. 2. Algorithm of spore dispersal from a source point

and contains 150 voxels. In average, about one third up to half of all voxels are processed by the function. Its complexity is $O(\#voxels)$.

The function `get_nodedist_list` considers each voxel in *vox_list* and calculates for all its leaves the distance to the source lesion. Then, it merges the created list with those of other voxels. Here, computation time is proportional to the number of leaves in the returned list `nodedist_list`. Our measures with our parameters indicate 41.7 % of all leaves are intercepted in average; so we admit that the size of `nodedist_list` is proportional to $\#leaves$. The time complexity is $O(\#leaves)$.

`quicksort` sorts *nodedist_list* according to the distance of leaves to the source lesion. This is done in $O(\#leaves \cdot \log(\#leaves))$ in average.

The `for` loop first tests for each leaf in *nodedist_list*, whether this leaf is indeed intercepted by the cone, using scalar product and trigonometric functions. It calculates the number of spores captured by the leaf according to its distance to the cone axis, and its potential to be infected according to its susceptibility decreasing exponentially with its age. At end, the leaf is possibly infected. The time complexity of the `for` loop is $O(\#leaves)$.

Theoretical time complexity of local dispersal is $O(\#leaves \cdot \log(\#leaves))$. However, elementary operations in the loop are quite complex and might require a lot of time.

During our simulation on a single grapevine, almost 4000 dispersal cones were thrown, spreading two to three millions of spores, of which hardly one million left the volume of the vine. Time spent during the different operations of the `source_dispersal` function was measured. For each operation corresponding to a step of this function and for all iterations, Tab. 1 reports the total time and the time of the longest execution.

Although $\#leaves$ is most of the time bigger than $\#voxels$, the function `get_nodedist_list` requires less time than `get_voxels_cone`. Indeed, the transition from a list of voxels to its list of leaves is a cheap operation. Computing the leaves intercepting cones without using voxels would take much more time.

Table 1. Comparison of execution times of the different operations in the function `source_dispersal`

Operations	Total time (s)	Longest execution time (ms)
`get_voxels_cone`	0.156	0.086
`get_nodedist_list`	0.087	0.095
`quicksort`	0.417	0.555
`for` loop	0.259	0.340

According to theoretical complexities, the last two steps are the most costly. The `quicksort` execution time is not expected to vary, whereas the processing of leaves in the `for` loop may increase with the model refinement, especially the possible multiple infections of a same leaf.

4 Description of the Parallel Simulator

The simulation of disease spread over a vineyard does not come to only simulate the disease on each grapevine. Indeed, there exist many interactions between stocks pertaining to the parasite dispersal, which will lead to communications. We decided to develop an SPMD [2] parallel code, so we must focus on data distribution and efficient communication strategies.

4.1 Data Distribution

As illustrated in Sect. 3, costly computational steps are host and pathogen growth and local dispersal of spores. Distributing stocks over processors makes the growth trivially parallel.

Local dispersion will generate communications between adjoining stocks allocated to different processors. So as to reduce these communications, the field should be cut out in blocks of maximal size and minimal common edges.

Moreover, grapevine vigour – parameter not taken into account currently – plays a role in plant growth and so in the number of organs. A vigorous stock area will produce an higher amount of computations. This point suggests to allocate a set of uniformly distributed stocks to a processor in order to privilege a good load-balancing.

The implemented load-balancing is static and consists in a 2D block-cyclic distribution. The need of a dynamic load-balancing will be adressed later. The set of stocks allocated to a processor is called its *local_stocks*.

4.2 Parallel Algorithm

Let us first describe precisely how the dispersal with neighbouring stocks and the long-range dispersal are modeled. Each processor has two matrices, named transmission matrix *TM* and dispersal matrix *DM*. Both have the field dimensions: they have as many lines as rows in the field, and as many columns as stocks

in the rows. Each one of their elements is associated with the stock of same coordinates in the field. An element in the transmission matrix is a list of dispersal cones transmitted to the corresponding stock by its neighbours. An element in the dispersal matrix is the number of spores received by the corresponding stock from all others in the vineyard.

After budding, an iteration on a processor is described in Fig. 3.

```
for vine in local_stocks do
  vine_computations(vine, day, T, ux, uy, uz, TM, DM)
end for
neighbouring_dispersal(local_stocks, day, T, TM, DM)
long_range_dispersal(local_stocks, DM)
```

Fig. 3. Algorithm of a parallel iteration after budding

`vine_computations` corresponds to the operations of Sect. 3 performed on a single grapevine, except that dispersal cones that exit the vine volume by lateral sides are added to the adequate cone lists of *TM*. Other exiting spores are accumulated in the *vine* data structure for later long-range dispersal.

`neighbouring_dispersal` sends the elements of the *TM* matrix using a `MPI_Alltoall` communication [3], and calls `source_dispersal` to propagate cones in grapevines. Exiting spores are not transmitted a second time to neighbouring stocks, they are all accumulated for long-range dispersal.

`long_range_dispersal` considers each grapevine in *local_stocks* and accumulates spores on the *DM* matrix entries by using Gaussian random drawings. Again, a `MPI_Alltoall` communication is performed on the *DM* matrix entries, and then spores are dispersed in grapevines thanks to the `source_dispersal` function. Currently, the value in *DM* associated to a stock is not only a number of spores, but n numbers corresponding to the amount of spores received by the stock from other ones in the n uniformly spread directions.

5 Performance Analysis

The implemented parallel simulator uses MPI [3] and MPI-communications within an SMP node are performed via shared memory. The distribution is block cyclic, and each block contains only one stock.

A platform located at CINES[1] (Montpellier, FRANCE) was used for the numerical experiments: it is a parallel cluster with 29 nodes of 16 IBM Power 3 processors. Up to 128 processors were used for simulations.

5.1 Load-Balancing Analysis

Currently load-balancing depends totally on the quality of the initial distribution. As we have seen, grapevine vigour can generate more computations on

[1] Centre Informatique National de l'Enseignement Supérieur

some areas of the field. The stocks that are primary foci of the epidemic can induce load-imbalance too.

However, three different periods could be distinguished during the simulation. During about the first 80 days preceding the budding, the computation cost is very low. This period is short in time and well-balanced. A large second period corresponds to the development of the plants and to the beginning of the epidemic. Some important load-imbalance can be observed at this moment, but this period does not represent the most costly part of the simulation. Because of the rapid disease spread, the last period contains most of the computations due to the dispersal of high numbers of spores and is rather well-balanced. So, the whole simulation balancing is determined by the one of that last period.

Simulations were performed on the Power 3 platform for a 32×32 field with several configurations: 2, 4, 8 and 16 processors on a same node (SMP), 32 processors on two nodes, 64 processors on four nodes and 128 on eight nodes. Table 2 reports for the days 150 and 220, belonging to the second and third periods of the simulation respectively, the maximum (top number) and the minimum (bottom number) times of each step over all processors. It provides also the communication times that include synchronization time.

Table 2. Maximum and minimum times in milliseconds of each step over all processors for 32×32 field simulations on the Power 3 platform

	day 150				day 220			
Number of processors	2	8	32	128	2	8	32	128
Computation time								
Vine growth	270 263	60 57.6	14.9 13.6	4.03 3.18	1593 1549	396 380	98.3 85.7	24.6 19.9
Local dispersal	52.2 19.5	17.4 4.75	8.62 1.16	7.75 0.27	15120 14670	3700 3526	981 813	266 183
Neighbour dispersal	29.7 0.27	11.6 0.2	7.88 0.24	8.51 0.5	6330 6215	1619 1430	426 336	121 64.3
Global dispersal	0.013 0.009	0.005 0.004	0.004 0.003	0.004 0.003	15760 15470	4044 3749	1049 892	295 195
Final dispersal	0.15 0.15	0.052 0.051	0.014 0.012	0.009 0.005	3468 3429	848 824	211 190	56.5 43.8
Communication time								
Neighbour dispersal	59.3 0.096	24.9 0.3	14.4 2.45	30.6 19.4	588 0.11	210 1.72	200 3.27	127 24.1
Global dispersal	30 0.55	12.1 0.69	12.7 5.02	66.5 56.5	402 0.7	338 0.62	185 4.79	126 20.1

A ratio of ten to several thousands can be noticed between step times at day 150 and those at day 220. The most costly computation at day 220 is rather well balanced up to 128 processors. Vine growth is clearly the fastest computational step and represents a very small part compared to the whole dispersal process. The minimum of communication steps corresponds to the effective communication time. The maximum measures in addition the idle time of the processor that ends first the computation. Furthermore, maximum communication time is small, if not negligible, in comparison with computation time, and it is decreasing with the number of processors at day 220.

5.2 Scalability Analysis

Maximum computation, communication and total times for the complete simulations are reported in Fig. 4 and other performance measures in Fig. 5.

Processors	2	8	32	128
Computation time	2234.4	553.7	141.4	37.76
Communication and idle time	70.8	41.1	25.9	21.89
Total time	2236	563.9	153.5	55.78

Fig. 4. Maximum computation, communication and total times in seconds for a complete simulation on a 32×32 field

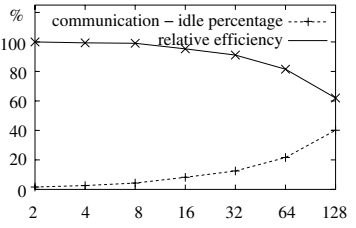

Fig. 5. Relative efficiency in comparison with 2 processors and ratio (communication and idle time) / (total time), for a complete simulation on a 32×32 field

As expected, computation time is about inversely proportional to the number of processors involved in the simulation, which is a good result.

Relative efficiency remains above 90 % up to 32 processors, it is about 80 % with 64 processors and drops to 63 % with 128 processors. Currently, there is no overlapping of communications by computations. It is only possible for the communications of the neighbour dispersal step and should be done in the future to improve performances. The ratio of communication time divided by global time increases from 2 to 128 processors because of load imbalance in computations. To enhance parallelism of the application the load-balancing should be refined.

6 Quality of the Current Biological Results

The simulator is still in the calibration phase and basic sub-models are being validated. Nevertheless, current outputs of the program are already coherent with field data. These outputs at field level consists of maps where each stock is represented by a point. The greyscale of this point depends on the severity of disease on the stock: from white to black, severity increases. Figure 6 represents three maps of a 32×32 field. Primary infection is located on four stocks in the top-left corner during days 110 and 127. The three maps correspond respectively to the days 160, 180 and 210. These results correspond qualitatively to field observations made by biologists.

7 Conclusion

These first results are encouraging: the simulator performances are quite good in terms of scalability and load-balancing.

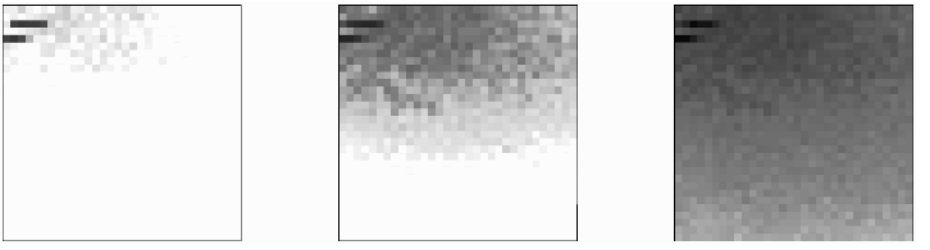

Fig. 6. Three maps of a 32×32 field at days 160, 180 and 210

The simulator is still in the calibration process and some future adjustments could strongly modify its behaviour. Our simple modeling of the vertical wind component is being reviewed with bioclimatologists. To take into account the important variations of this component [5], several dispersal cones will be spread from each sporulating colony every day. Each cone will carry a part of the spores released by the colony during the given day. This modification will increase the computational time of local dispersal and will generate more communications between neighbouring stocks.

We also consider the extension of the local dispersal domain to all the neighbouring stocks of a source stock, including its neighbours in the previous and next rows.

Imbalance may appear when taking into account the grapevine vigour, hence it could imply to improve load-balancing strategy.

Moreover, it will be interesting to model prophylactic[2] methods.

References

1. Bulit (J.) et Lafon (R.). – Powdery mildew of the vine. *In : The powdery mildews*, éd. par Academic press, London. – DM Spencer, 1978.
2. Grana (Ananth), Gupta (Anshul), Karypis (George) et Kumar (Vipin). – *Introduction to Parallel Computing*. – Addison Wesley, 2003, second edition édition. ISBN 2-201-64865-2.
3. MPI Forum. – *Message Passing Interface MPI Forum Home Page*. Available from `http://www.mpi-forum.org/`.
4. Naulin (J.-M.), Tessier (G.), Bailey (D.), Langlais (M.) et Calonnec (A.). – A host/pathogen simulation model : Powdery mildew and vine. – february 2005. Submitted to New Phytologist.
5. Raupach (M. R.), Finnigan (J. J.) et Brunet (Y.). – Coherent Eddies and Turbulence in Vegetation Canopies: The Mixing-Layer Analogy. *Boundary-Layer Meteorology*, vol. 78, 1996, pp. 351–382.
6. Samet (Hanan). – *The Design and Analysis of Spatial Data Structures*. – University of Maryland, Addison-Wesley Publishing Company, 1989. ISBN 0-201-50255-0.

[2] set of measures to fight against epidemics; for example chemical treatments.

7. Shigesada (Nanaka) et Kawasaki (Kohkichi). – Invasion and the range expansion of species: effects of long-distance dispersal. *In : Proceedings of BES Annual Symposium 2001 'Dispersal'*, chap. 17, pp. 350–373. – Blackwell Science (in press), 2002.
8. Whitted (Turner). – An improved illumination model for shaded display. *Communications of the ACM*, vol. 23, n° 6, 1980, pp. 343–349. – ISSN:0001-0782.
9. Zawolek (M. W.) et Zadocks (J. C.). – Studies in Focus Development: An Optimum for the Dual Dispersal of Plant Pathogens. *Phytopathology*, vol. 82, n° 11, 1992, pp. 1288–1297.