Ecological Modelling, 58 (1991) 159–183
Elsevier Science Publishers B.V., Amsterdam

Spatial ecosystem modelling using parallel processors

Robert Costanza and Thomas Maxwell

Chesapeake Biological Laboratory, Center for Environmental and Estuarine Studies, University of Maryland, P.O. Box 38, Solomons, MD 20688, U.S.A.

(Accepted 1 May 1991)

ABSTRACT

Costanza, R. and Maxwell, T., 1991. Spatial ecosystem modelling using parallel processors. *Ecol. Modelling*, 58: 159–183.

We have developed a spatial modeling workstation, that consists of a combination of hardware and software tools that allow development, implementation and testing of spatial ecosystem models in a convenient desktop environment. The system links commercially available Geographic Information Systems (GIS) for managing spatial data with a commercially available general dynamic simulation system for developing unit models (STELLA^{TM}) and a Spatial Modeling Package (SMP) that we developed for linking the unit models into a spatial array, handling horizontal exchanges, and running the array as a spatial model. The spatial model code is executed on either: (1) transputers (parallel processors) resident in a desktop microcomputer; or (2) on a remote Connection Machine parallel mainframe computer with 64 K processors. Resulting time series maps are readable by the GIS system for further display and analysis.

In this paper we: (1) describe the hardware and software system; (2) describe a hypothetical model of simple diffusion over a landscape that we developed and tested using the system; and (3) describe a practical application of the system to spatial modeling of long-term habitat succession in the coastal Louisiana region. We find that a system using eight transputers on a Macintosh IIci can run spatial ecosystem models in about the same time as a CRAY X/MP.

INTRODUCTION

Spatial modeling of ecosystems is essential if one's modeling goals include developing a relatively realistic description of past behavior and predictions of the impacts of alternative management policies on future ecosystem behavior (Risser et al., 1984; Costanza et al., 1990; Sklar and Costanza, 1991). Development of these models has been limited in the past by the large amount of input data required and the difficulty of even large mainframe serial computers in dealing with large spatial arrays. These two limitations have begun to erode with the increasing availability of remote sensing data and GIS systems to manipulate it, and the development of parallel computer systems that allow computation of large, complex spatial arrays. Improved computational capabilities (both in terms of raw speed and ease of use) can benefit many aspects of spatial modeling and landscape ecology. In efforts to describe landscapes, many new, computationally intensive indices have been developed (Colwell, 1974; Mandelbrot, 1977, 1983; Weins et al., 1985; Gardner et al., 1987; Kummel et al., 1987; Urban et al., 1987; Costanza, 1989; Turner et al., 1989) that could benefit from faster and easier to use computer systems. In this paper we describe how far these computer developments have come and how they may now be used in building and running dynamic spatial ecosystem models.

We define a dynamic spatial model as any formulation that describes the changes in a spatial pattern from time t to a new spatial pattern at time t + 1, such that

$$X_{t+1} = f(X_t, Y_t)$$

where $X(t)$ is the spatial pattern at time t and $Y(t)$ is a set of array, vector or scalar variables that may affect the transition (Sklar and Costanza, 1991). There are four major disciplines that use spatial models within the broad topic of ecology: geography, hydrology, biology and ecosystem science. Within geography we can further subdivide spatial models into geometric, demographic and network models. In hydrology there are basically two types of spatial models of fluid dynamics: finite element models, or hydrodynamic models, and finite difference models, or general circulation models. Spatial models in biology include growth models, population models, and point-averaged ecosystem models. Spatial ecosystem models can be classified as stochastic landscape models or process-based landscape models. For a recent review of spatial modeling, see Sklar and Costanza (1991).

Although all of the forms of dynamic spatial modeling described above are highly amenable to parallel processing, our primary focus in this paper is on process-based landscape models. These models simulate spatial structure by first compartmentalizing the landscape into some geometric design and then describing flows within compartments and spatial processes between compartments according to location-specific algorithms. Examples of process-based, spatially articulate landscape models include wetland models (Sklar et al., 1985; Costanza et al., 1986; Kadlec and Hammer, 1988; Boumans and Sklar, 1991; Costanza et al., 1990), oceanic plankton models (Show, 1979), coral reef growth models (Maguire and Porter, 1977), and fire ecosystem models (Kessell, 1977).

The Coastal Ecological Landscape Spatial Simulation (CELSS) model, for example, is a process-based spatial simulation model consisting of 2479 interconnected cells, each representing 1 km^2, constructed for the

Atchafalaya/Terrebonne marsh/estuarine complex in south Louisiana (Sklar et al., 1985; Sklar and Costanza, 1986; Costanza et al., 1990). Figure 1 shows the location of this study area and of the Barataria study area referred to later on. Each 1 km^2 cell in the CELSS model contains a dynamic, non-linear ecosystem simulation model with seven state variables, similar to the one shown in Fig. 2. The model is generic in structure, and can represent one of six habitat types by assigning unique parameter settings. Each cell is potentially connected to each adjacent cell by the exchange of water and materials. The volume of water crossing from one cell to another is a function of water storage and connectivity. Connectivity is a function of landscape characteristics, including habitat type, drainage density, waterway orientation, and levee height. Habitat succession occurs in a cell when its environmental variables (e.g., salinity, elevation, water level, productivity, etc.)

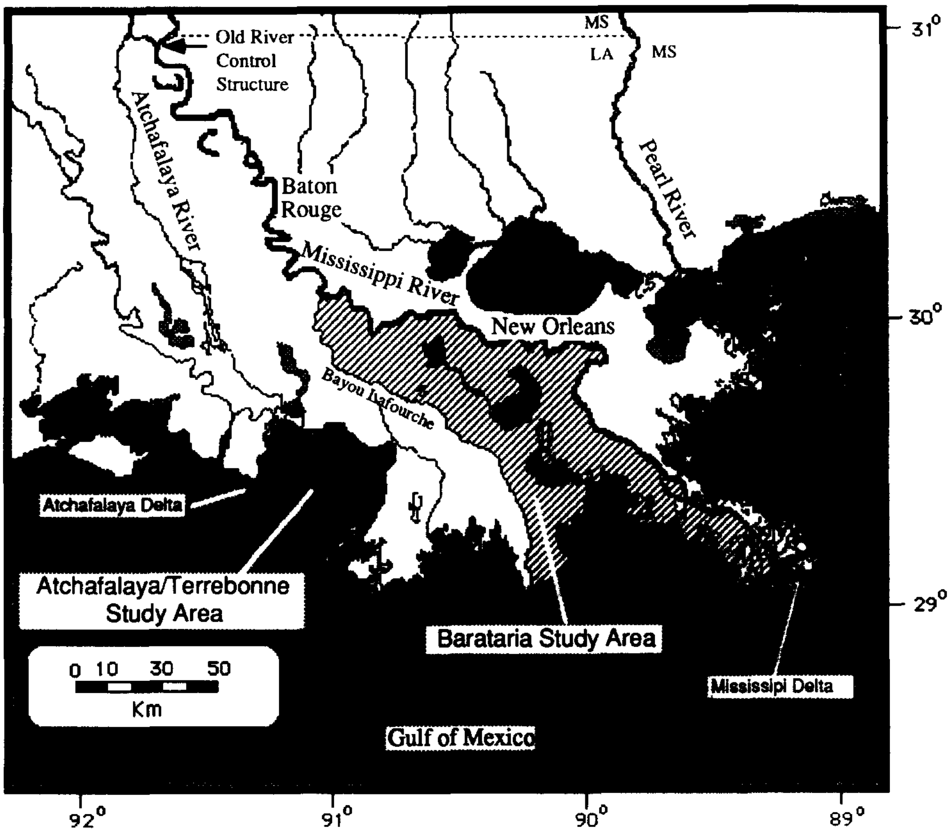


Fig. 1. Map of southern Louisiana showing the Atchafalaya/Terrebonne and the Barataria study areas, Louisiana, USA.

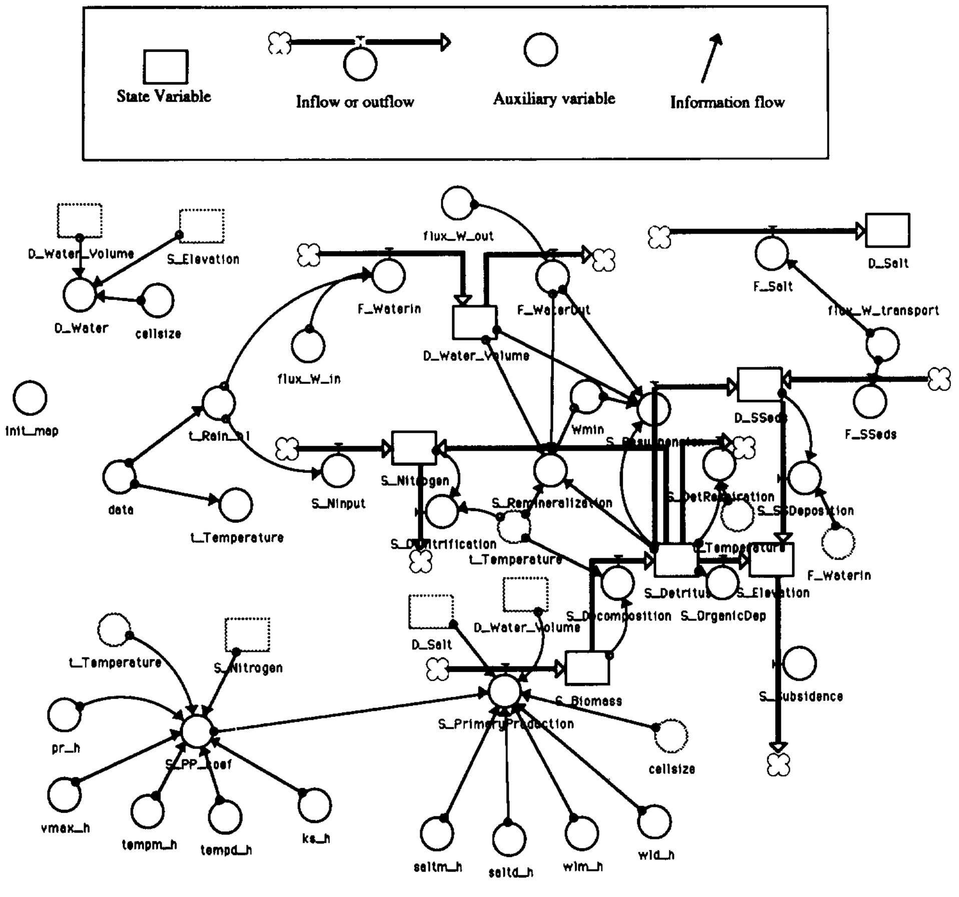


Fig. 2. STELLA diagram of the generic unit model used for both the Atchafalaya/Terrebonne and the Barataria modeling studies.

fall outside the ranges for its designated habitat type. Succession means that the cell habitat type and all the associated parameter settings are switched to a set more adapted to the changed conditions. For example, if salinity in a cell that was initially fresh marsh goes beyond a threshold value of 3 ppt and remains at this high level for more than 45 weeks then the cell is converted to brackish marsh with all its associated parameters including a habitat specific function for primary production. The CELSS model applied to the Atchafalaya/Terrebonne area of coastal Louisiana explained about 90% of the 1978 ecosystem type calibration data and predicted about 79% of the 1983 verification data (Costanza et al., 1990). Figure 3 shows some sample output indicating the real and simulated habitat changes from 1956 to 1983, along with nitrogen and water predictions from the model. Table 1 indicates the range of management options and past and future climate scenarios that the model has been used to analyze.

The CELSS model was written in standard FORTRAN (3021 lines of code) and was run with a time step of 1 week on a variety of computers including a VAX 11/780, IBM 3034, and CRAY X/MP. A typical 22-year run at a weekly time step (1144 total time steps) for all 2479 cells with eight state variables each (19 832 total simultaneous difference equations) takes about 24 h of CPU time on the VAX, 2 h on the IBM, and 15 min on the CRAY. Most of the full scale runs of the model were done on the CRAY with the other computers used mainly in the early model development stages.

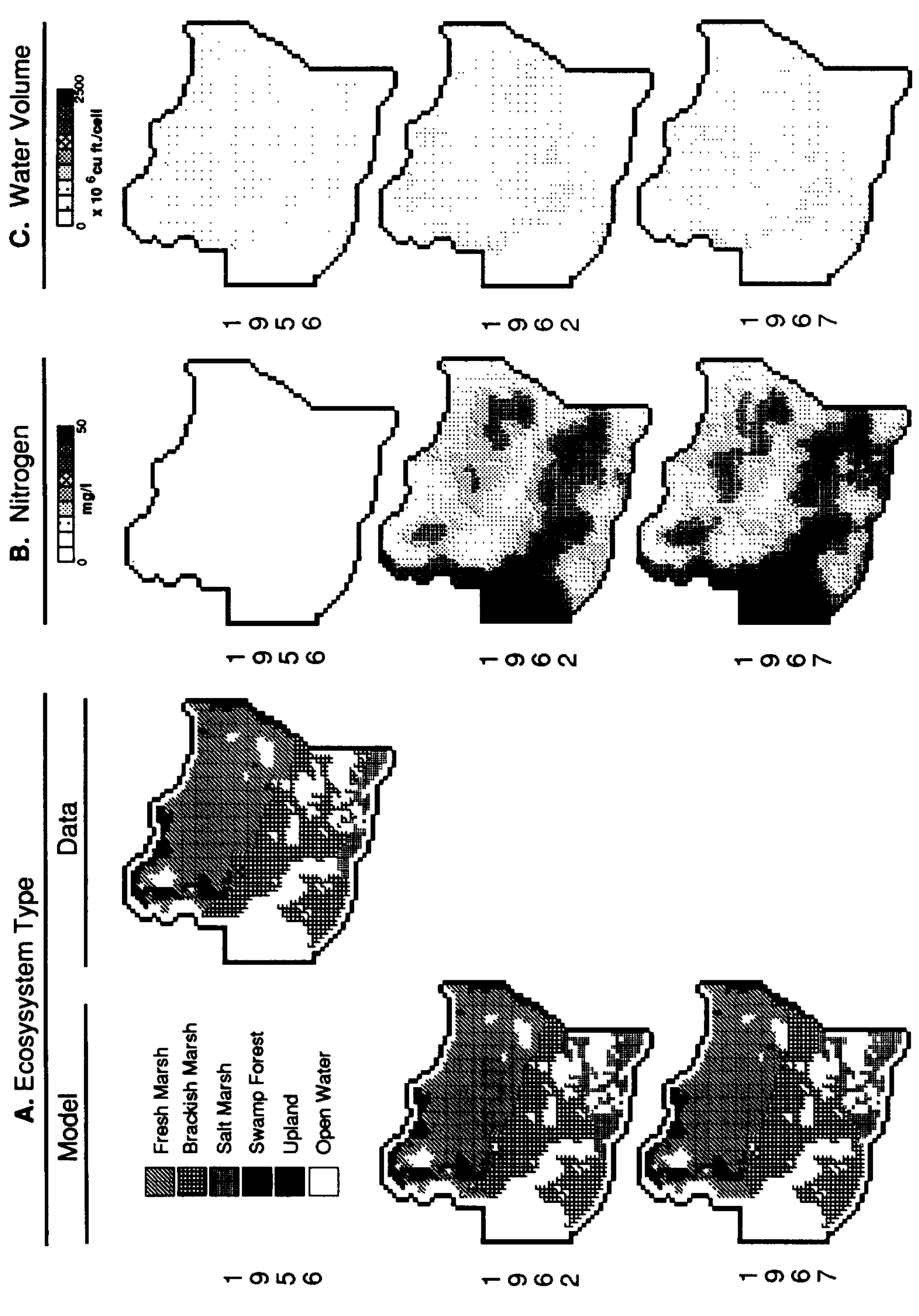
The original CELSS model took four people about 4 years (16 person-years) to fully develop and implement using a supercomputer. The model has proved to be very effective at helping us understand complex ecosystem behavior and at guiding policy and research (Costanza et al., 1990). We are now concerned with reducing the time involved for both developing and running this type of model, and moving the modeling to smaller, less expensive computers. Toward that end we have developed the integrated spatial modeling workstation described below.

SPATIAL MODELING WORKSTATION DEVELOPMENT

Computer systems are much more effective tools for research and education if they are easy to use. In order to be effective management tools, spatial models with adequate resolution (like the CELSS model) have until recently required supercomputers. Our goal was to utilize new developments in parallel computer architectures to bring useful spatial ecosystem modeling to an easy-to-use microcomputer platform. Toward this end we developed a spatial ecosystem modeling system to run on Apple Macintosh II™ computers (with the possibility of exporting code to run on a Connection Machine for very large problems). The system consists of three major software components linked together using Hypercard™. The system is shown diagrammatically in Fig. 4. The major phases of developing a spatial model are: (1) unit model development and testing; (2) data assembly and manipulation; and (3) lining the data and unit models to run in space and time. In our system the unit model development and testing is done using STELLA™, a general, commercially available*, dynamic simulation model development package that is very easy to learn and use (Costanza, 1987). The data assembly and manipulation is done using one of two GIS: GRASS** for

*High Performance Systems Inc., Lyme, New Hampshire.

**Geographical Resource Analysis Support System, developed by the U.S. Army Corps of Engineers.


A. Ecosysystem Type
Model
Data
Fresh Marsh
Brackish Marsh
Salt Marsh
Swamp Forest
Upland
Open Water
B. Nitrogen
0
50
mg/l
C. Water Volume
0
2500
x 10 6 cu ft./cell
1956
1962
1967

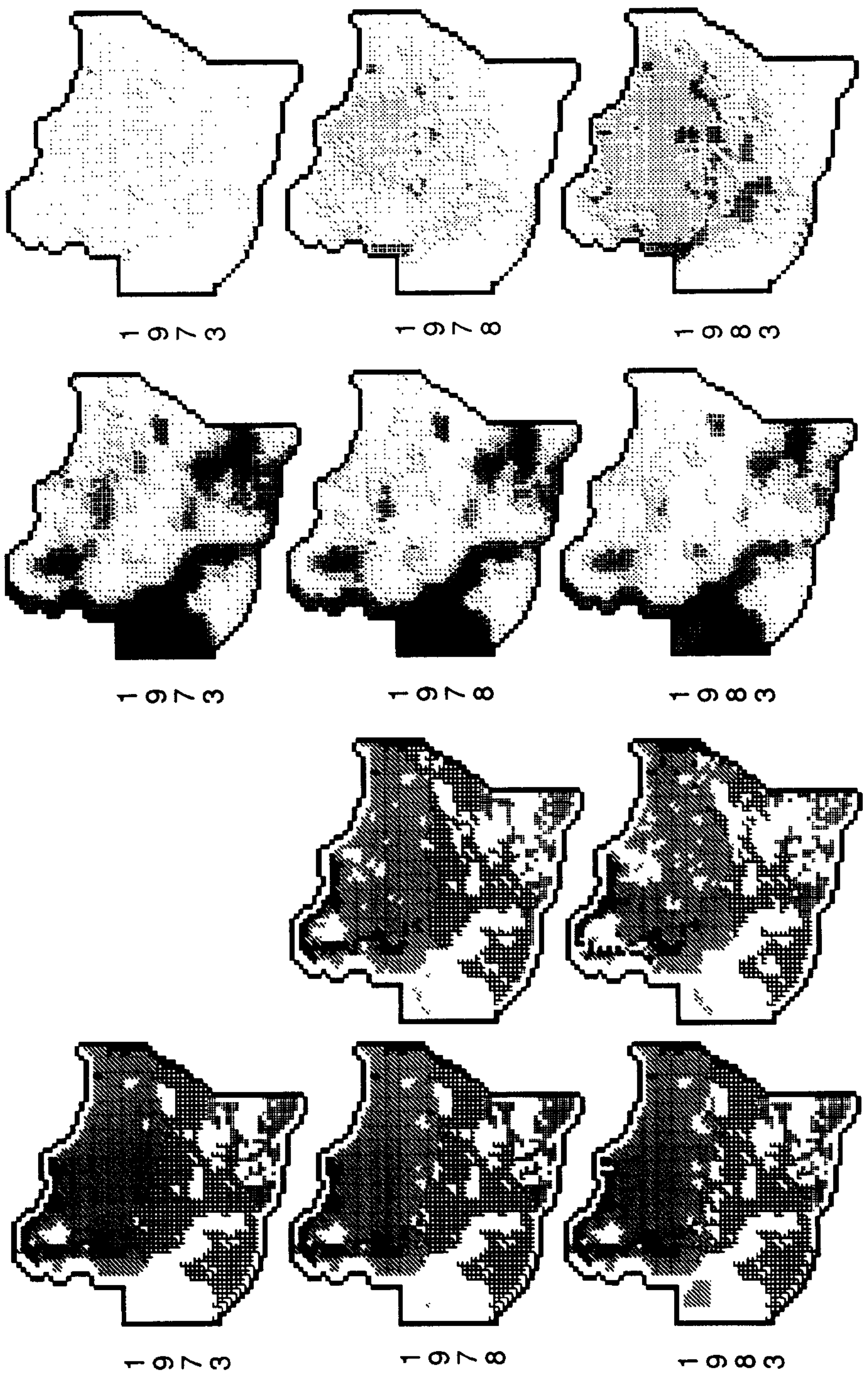


Fig. 3. Sample model output and data from the calibration phase of the CELSS model application to the Atchafalaya/Terrebonne study area (Costanza et al., 1990).

TABLE 1

Number of square kilometers of each ecosystem type for the three years for which data is available, and for the year 2033 for various scenarios from the CELSS model (Costanza et al., 1990). Changes from the base case (in km^2) are indicated in parentheses

	Swamp	Fresh Marsh	Brackish Marsh	Saline Marsh	Upland	Total Land	Open Water
1956	130	864	632	98	13	1737	742
1978	113	766	554	150	18	1601	878
1983	116	845	347	155	18	1481	998
2033 Scenarios[a]:							
Climate scenarios[b]:							
Climate run 3 (climate base case)	84	871	338	120	10	1423	1056
Climate run 1	79 (-5)	874 (+3)	337 (-1)	127 (+7)	10 (0)	1427 (+4)	1052 (4)
Climate run 4	85 (+1)	900 (+29)	355 (+17)	130 (+10)	10 (0)	1480 (+57)	999 (-57)
Climate run 5	83 (-1)	891 (+20)	332 (+6)	126 (+6)	10 (0)	1442 (+19)	1037 (-19)
Mean Climate	94 (+10)	974 (+103)	402 (+64)	136 (+16)	11 (+1)	1617 (+194)	862 (-194)
Weekly Average Climate	128 (+44)	961 (+90)	813 (+475)	300 (+180)	11 (+1)	2213 (+790)	266 (-790)
Management scenarios:							
No levee extension (base case)	100	796	410	123	15	1444	1035
Two reach levee extension	98 (-2)	804 (+8)	399 (-11)	123 (0)	15 (0)	1439 (-5)	1040 (+5)
CLF Marsh Management	102 (+2)	798 (+2)	409 (-1)	123 (0)	15 (0)	1447 (+3)	1032 (-3)

Falgout Weir	104 (+4)	799 (+3)	403 (-7)	122 (-1)	16 (+1)	1444 (0)	1035 (0)
Full six reach levee extension	103 (+3)	790 (-6)	362 (-48)	122 (-1)	15 (0)	1392 (-52)	1087 (+52)
Fresh water diversion (FWD)	103 (+3)	803 (+7)	404 (-6)	123 (0)	15 (0)	1448 (+4)	1031 (-4)
FWD and Palmetto Weir	102 (+2)	802 (+6)	407 (-3)	123 (0)	15 (0)	1449 (+5)	1030 (-5)
FWD and Superior Weir	104 (+4)	799 (+3)	404 (-6)	123 (0)	15 (0)	1445 (+1)	1034 (-1)
FWD, Superior and Palmetto Weirs	104 (+4)	792 (-4)	407 (-3)	123 (0)	15 (0)	1441 (-3)	1038 (+3)
FWD, Superior and Falgout Weirs	104 (+4)	803 (+7)	407 (-3)	122 (-1)	15 (0)	1451 (+7)	1028 (-7)
Boundary scenario[c]:							
EPA low sea level rise	104 (+4)	800 (+4)	411 (+1)	124 (+1)	15 (0)	1454 (+10)	1025 (-10)
EPA high sea level rise	89 (-11)	794 (-2)	396 (-14)	131 (+8)	15 (0)	1425 (-19)	1054 (+19)
Historical scenarios[d]:							
No original Avoca Levee	84	951	350	126	13	1524	955
No impacts	130	863	401	144	12	1550	929

[a]The summary maps and this table indicate the "dominant" habitat type for each cell (i.e., the ecosystem type that was present in the cell for the largest amount of time during the year). Alternately, we could have added the total number of cells of each ecosystem type for each week of the simulated year and divided the totals by 52. While this gives a somewhat more accurate picture of the habitat distribution it is inconsistent with the totals from the maps.

[b]The climate analysis scenarios used a slightly different set of parameters for the model than the other scenarios. See text for details.

[c]EPA low scenario is 50 cm rise by the year 2100. We used 0.46 cm/year which is double the historical rate of eustatic sea level rise in the study area of 0.23 cm/year. Subsidence in the study area varies horizontally from 0.57 to 1.17 cm/year, giving historical rates of apparent sea level rise (eustatic rise plus subsidence) of 0.8–1.4 cm/year. EPA high scenario is 200 cm rise by the year 2100 (1.67 cm/year eustatic or 2.24–2.84 cm/year apparent). Base case for comparison was the no levee extension case.

[d]No comparisons with a base case are given for the historical scenarios since these runs started in 1956 rather than 1983.

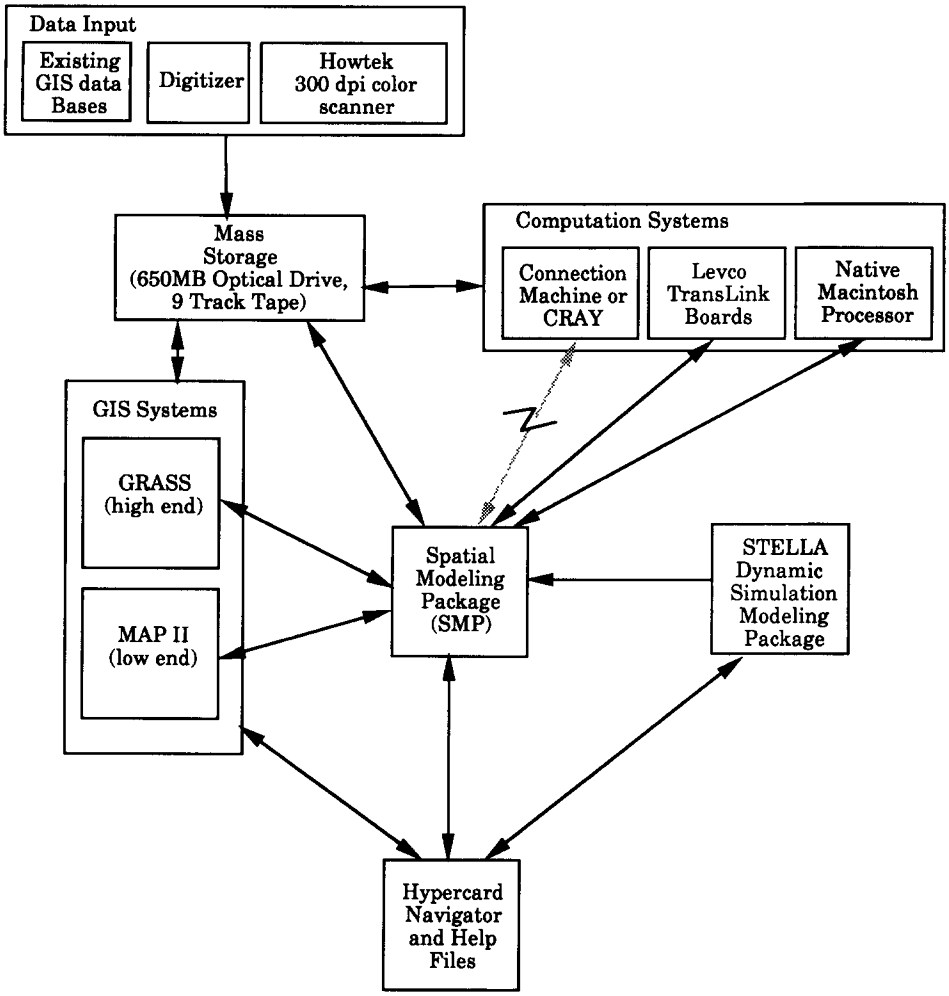

Data Input
Existing GIS data Bases
Digitizer
Howtek 300 dpi color scanner
Mass Storage (650MB Optical Drive, 9 Track Tape)
Computation Systems
Connection Machine or CRAY
Levco TransLink Boards
Native Macintosh Processor
GIS Systems
GRASS (high end)
MAP II (low end)
Spatial Modeling Package (SMP)
STELLA Dynamic Simulation Modeling Package
Hypercard Navigator and Help Files

Fig. 4. Diagram of major hardware and software components of the spatial modeling workstation.

more elaborate and complex operations, or MAP II^{TM*} for more moderate applications. These pieces are linked together with a Spatial Modeling Package (SMP) which we have developed. The SMP prepares code to be run on either the Connection Machine parallel computer, Levco transputer boards installed in the Mac II**, or the native Macintosh II series 680 × 0 processors. These systems generate output files that can be read and manipulated using one of the GIS systems.

*Available from John Wiley and Sons, Inc.
**Available from Levco Inc., San Diego, CA.

The system can be used either as four separate software modules or through a Hypercard front end. Hypercard allows easy navigation through the system, and online help and explanation. Using the Hypercard^{TM} front end, even first time users can gain access and effectively use the system.

PARALLEL COMPUTERS

Most existing computers are 'serial'. They have one central processor and all operations must be done one after the other, in series, using the single processor. There are inherent speed limitations with this kind of computer architecture, even with the fastest processors. However, many kinds of problems (but not all) can be separated into a number of tasks that can all be performed simultaneously. The CELSS spatial ecosystem model (Costanza et al., 1990), image processing, neural networks, and GIS systems (Burrough, 1986) are good examples. For these kinds of problems, much faster solutions on much less expensive hardware are *theoretically* possible by using 'parallel computers' with many processors. While this idea has been around for some time, it was hindered in the past by the high cost of central processors and the increased complexity and lack of availability of software for parallel machines. With the advent of microcomputers and low cost processors, the parallel approach is now beginning to be seriously implemented. Several parallel computing options are currently available and there is much work in progress on parallel software systems.

We investigated two basic approaches to parallel computer implementation of spatial ecosystem models:

1. The Connection Machine, manufactured by Thinking Machines Inc., which is a large, massively parallel (64 000 processor) 'mainframe' parallel computer.
2. Transputer based microcomputer systems. We performed a preliminary assessment of the available hardware and software and a more in-depth analysis and review of two systems available for the Apple Macintosh II computer: Levco's TransLink^{TM} transputer boards and the Chorus^{TM} networkable transputer system manufactured by Human Devices Inc.

Details of these systems are given below. After our review, we decided to continue work on the Connection Machine for very large scale models and to use the Levco transputer boards for microcomputer applications.

Connection Machine

The Connection Machine (CM) is a large, expensive 'mainframe' implementation of the parallel processing idea that can have up to 64 000 processors. We are currently using a CM at the University of Maryland's College Park campus that is a 'quarter machine', with 16 000 processors. The CM architecture is perfectly suited for spatial ecosystem simulation models. Each spatial cell in the model can be assigned to its own processor and the entire array can be run in parallel in only slightly more than the time it would take to run a single cell.

Access to the CM is through a 'front end' serial computer, typically a VAX running ULTRIX. Both parallel and serial code reside in the memory of the front end. Data can reside in either the front end memory or in the memories of the individual processors of the CM. Typically, scalar or 'mono' data will be stored on the front end, whereas parallel variables will be spread out over the processors of the CM.

All code development, editing, debugging, compiling, linking and running is done in the familiar environment of the front end computer. The CM behaves as an extra processor connected to the VAX. Serial code is executed by the front end in the usual fashion; parallel instructions are broadcast to all data processors in the CM simultaneously.

A high speed communication network called the 'router', connects every processor with every other processor. The router allows every processor to send a message to any other processor, all at the same time, with a total throughput exceeding 3 gigabits/s. In addition there is a local network called the NEWS grid for very fast local communication over N-dimensional grid topologies.

If more processors are required than are physically available, the CM also has the capacity of supporting 'virtual processors'. By dividing the memory of each processor into N portions, each processor can be made to simulate N processors. Since the N processors are activated sequentially, run time is increased by a factor of N. Changing the size of a program or the number of physical processors available requires no code changes (except perhaps to redefine array sizes). Ultimately the number of (virtual) processors available is limited only by the amount of available memory.

The CM programming languages are parallel extensions of FORTRAN, Lisp, and C++ . The parallelization format varies among languages, but generally is based on the concept of parallel variables ('pvars') that can be envisioned as arrays with one component stored in the local memory of each processor. Instructions broadcast from the front end result in the execution of operations on each component in parallel. For example, if A and B are pvars, then the result of the instruction '$C = A + B$' is the addition of the local components of A and B in each processor, and the storage of the sum in pvar C. This parallel code, which is identical in form to serial code, has resulted in NP additions taking place in parallel, where NP is the number of selected processors.

We have done preliminary exploratory simulations of spatial models on the CM computer and have been very impressed with the results. We plan to continue to use the CM for both theoretical development of parallel simulations and to run large spatial arrays quickly. Because it is a mainframe system, however, the CM does not allow as high a degree of user interaction as we would like. We are therefore also pursuing (in parallel, of course) completely microcomputer-based parallel computer solutions.

Levco TransLink transputer boards

The Levco TransLink™ system consists of TransLink cards that plug into the slots of a Macintosh II series computer (similar boards are also available from other manufacturers for IBM PC compatible computers). Each card can hold up to four Inmos™ Transputers, each with 256 kB, 1 MB, or 4 MB of memory. The Transputer (transistor-computer) is a 32-bit, high speed reduced instruction set computer (RISC) based on VLSI technology manufactured by Inmos, Ltd. in the U.K.

The TransLink system operates as an extension of the native Macintosh system. The transputer code is written in C with parallel extensions and developed under the Macintosh Programming Workshop (MPW). Most of the code in the parallel program is identical to the serial version. The steps of the parallelization process (described below under 'SMP Grid Manager') are, for the most part, handled by one or two function calls. The process of coordinating communication between the two processors is somewhat complicated, but examples exist for the most common topologies which can be adapted for particular applications.

According to Levco literature, 'Programs that have little or no vector code*, that can be made to run in parallel, and that require little communication between processes like: finite state analysis, Monte-Carlo simulation, ray tracing, image processing, differential equations, etc., would require 6–8 modules on 2 TransLink cards to match the power of the CRAY in these instances.' CELSS-type spatial simulation models (Costanza et al., 1990) fit this definition perfectly. One should therefore be able to run spatial ecosystem models on a suitably equipped Macintosh II in about the same amount of CPU time as on the CRAY XM/P. Actual timing results are reported below. Since a local microcomputer would eliminate time intensive

*Vector code refers to computations that can be converted into vectors and operated on in a 'pseudo-parallel' way by treating the elements of the vectors as being in a 'pipeline'. Serial elements in the pipeline can be manipulated simultaneously. Most non-linear models and those with discontinuities do not vectorize well.

communication with the CRAY, the real time necessary to do the job would be significantly less on the microcomputer. The total cost of an eight transputer system (about $13,000 excluding the Mac II itself) is modest indeed for a system with such capabilities (especially considering the $10 million cost of a CRAY).

STELLA MODELLING PACKAGE

The individual cell models are developed using STELLA, a software package for the Macintosh designed to facilitate model building (Costanza, 1987). It is not *necessary* to use STELLA for this purpose since all that is required to send to the next component of the system is a set of difference equations and there are many ways of arriving at these equations. But STELLA greatly facilitates this process and it is just this sort of computer facilitated improvement in speed and ease of use that we are after in the overall system. STELLA uses symbols that are based on Jay Forrester's systems dynamics language (Forrester, 1961) that has become popular among modeling practitioners as a way to define and communicate a model's structure. These icons, representing stocks, flows, and functional relationships, are manipulated with the mouse to graphically build the model structure. Once the structure is established, the initial values for the state variables and the form of the rate equations are defined by clicking on the appropriate icon to generate a dialog box. The defining equations can then be typed in analytically, making use of numerous built-in functions, or entered graphically, using either a graph pad or a data table. When the definitions are complete, the model can be run. STELLA will scale the variables automatically and plot up to four variables simultaneously on each output 'page'. As many pages as necessary can be used to view as much of the model output as desired. In addition there is the option to plot the change over time of one variable against the change of another variable. STELLA greatly increases the ease with which one can change the model and see the effects of those changes on the model's behavior. It allows the computer to handle the computational details (which is, we think, as it should be) and frees the user to concentrate on modeling, greatly reducing model development time.

The STELLA models represent local ecosystem site models which can be linked through horizontal flows of nutrients, sediments, etc. to form spatial ecosystem models. STELLA will generate an output file containing a set of difference equations which describe the model. This equation file includes initial conditions, model structure equations representing the dynamics of the state variables, and auxiliary equations describing the parameters. These files serve as input to the SMP translator, which integrates the various local models into a spatial ecosystem model.

GIS SYSTEMS

Geographic Information Systems (GIS) are software systems used to collect, store, and manipulate spatially referenced data. The development of spatial ecosystem models requires access and manipulation of large quantities of spatial data, such as land use, habitat, and climate maps. These maps will typically be collected from many different sources and come in widely differing formats. The GIS component of the SMP handles the crucial tasks of translation, manipulation, and storage of this data.

The SMP can utilize several GISs, including MIPS (Map and Image Processing System), GRASS (Geographic Resource Analysis Support System), and MAP II (Map Analysis Package II). MIPS, which runs on the IBM PC, and GRASS, which runs on the Mac II under A/UX, are large scale systems capable of data digitizing, data read-in and read-out conversions, image processing, data analysis, and data presentation. These two systems have the advantage of being powerful, versatile, programmable, and capable of performing translation between various map formats. GRASS is capable of reading data in the digital line graph, digital terrain elevation, digital elevation models and several other formats. Their disadvantages are that they are difficult to learn and use, and are not directly accessible from the Macintosh Operating System (Mac OS).

MAP II, which runs under the Mac OS, is a simple, easy to use, raster-based GIS. It is adequate for most applications, but lacks (in the current version) the flexibility and extensibility of GRASS and the other systems. In general, we have tried to make the Spatial Modeling Workstation work with a number of GISs so that one is not limited to those mentioned above, but can take advantage of new developments as they arise.

SPATIAL MODELING PACKAGE (SMP)

The SMP consists of three components: (1) a *translator* for converting STELLA equations into parallel C code; (2) a *grid manager* for setting up and running the spatial array on the appropriate parallel computer; and (3) a *GIS interface* for handling input and output from the GIS systems. These components are described in more detail below.

SMP translator

The core of the SMP is a translator module which converts STELLA equation output files to C code that can utilize either the Connection Machine parallel computer, Levco transputer boards installed in the Mac II, or the native Macintosh II series 680 × 0 processor. This module is written in

THINK C and runs as an application on the Macintosh family of computers.

The first step of the translation process involves converting the STELLA equations into C code. Some of the steps involved in this process are: (1) identifying and defining variables; (2) identifying and translating the difference equations; (3) inserting semi-colons and 'for' loops; (4) translating the STELLA 'if-then' statements to their C equivalents; (5) translating STELLA supplied functions to their C math library equivalents; and (6) defining graph functions to represent the graphically defined STELLA variables. The translated file will run on the Macintosh under THINK C.

The next step involves converting the serial C code of the STELLA model, which represents a single ecosystem site model, to parallel C code that will access either the transputer modules or the Connection Machine parallel supercomputer. The parallel model is constructed as a 2D spatial array of cells, each of which contains an ecosystem site model, and represents a particular spatial area. The key step of the translation process involves translating scalar variables to 2D arrays, such that each component of the array represents the value of that variable in the corresponding site model. Thus if we are working with an N by N grid, Biomass $\rightarrow$ Biomass[N] [N]. When the parallel code is run on the Connection Machine, one processor is assigned to each grid cell. When the code is run on the Mac II with NT transputer modules, each transputer handles a unique sub-array covering $1/NT$ of the total grid.

The next step in the parallelization process involves the implementation of inter-grid cell communication. The spatial models described here require only nearest neighbor communication among grid cells, which greatly simplifies this step. Inter-processor communication format is dependent on both the type of parallel machine and the parallel programming language being utilized. For the case in which the grid dynamics are divided among several transputers, the form of inter-grid-cell communication will depend on the location of the grid cells. If both cells are allocated to the same processor, communication is a simple array reference. If the cells are allocated to different processors, additional code must be supplied to handle the transfer of data between processors. These transfers will be required along all the boundaries at which the sub-arrays allocated to different processors connect.

In the process of parallelizing serial C code, variables are handled differently according to the type of inter-cell communication required in their dynamics. We utilized naming conventions within STELLA to distinguish these classes of variables. First, variables can be classified as either mono or parallel. Mono variables remain constant over the grid (but may vary over time) while parallel variables can take on different values at different grid points. They may be constant or variable over time. Mono variables have no special naming convention; parallel variables must begin with 'Sus_', 'Dus_', or 'Fus_'. Thus we designate three classes of parallel variables: static (S), dynamic (D), and flux (F). Parallel static variables are state variables whose value is independent of intercellular flows. Parallel dynamic variables are state variables whose value depends on intercellular flows. Parallel flux variables are flow variables between grid cells. Static variables parallelize without complication. For each dynamic variable we must create special code to handle the interprocessor communication. Flux variables take on special values that represent different intercellular flow types.

We defined a set of basic flows that can be utilized in defining intercellular fluxes. For example, in order to establish a given flux, which we will denote as w0, among the cells of the parallel salt variable 'D_salt', we define (from within STELLA) the salt flux variable 'F_salt' and set it equal to the name of the predefined flux function 'flux_w0'. When translated, this code generates the parallel function call which sets up the appropriate intercellular fluxes. All fluxes must be either taken from the predefined set or programmed by hand in the parallel code.

Table 2 shows an example of a simple set of difference equations as they appear in STELLA format and in parallel C format.

SMP grid manager

The SMP translator creates a set of functions that are called from the 'grid manager', a generic parallel processing program. This program handles the details of the process of simulating spatial dynamics using transputers. The set of tasks that it performs include: (1) reading data and parameter files; (2) accessing transputer system information; (3) setting up grids and coordinates; (4) decomposing the processors over the grid such that a sub-grid is allocated to each processor; (5) assigning addresses to neighboring processors for use in communication calls; (6) initializing the graphics system and defining viewports such that each processor has a region of the screen corresponding to its sub-area of the grid; (7) assigning grid boundary values; and (8) setting up information exchange along the adjacent boundaries of sub-grids assigned to neighboring processors. Since the grid manager accesses transputer system information directly and decomposes the processors over the grid accordingly, the code is independent of the transputer hardware configuration. Modules can be added or subtracted and the grid manager will adjust accordingly.

We have tested the grid manager and generated timing comparisons with a simple reaction-diffusion program. This program simulates the dynamics of a single state variable 'phi' defined on a 60 by 60 grid (see below for a complete description of this program). The dynamics consist of a diffusion term plus a non-linear function of phi. The run time required for 100 iterations

TABLE 2

STELLA equations and the corresponding parallel C equations

Language	Equations
STELLA	see STELLA equations below
Parallel C	see Parallel C equations below

STELLA

```
{Initialization equations}
INIT(D_Phi) = 10
D_Births = .15*D_Phi
D_deaths = .00015*D_Phi*D_Phi

{Structure equations}
D_Phi = D_Phi + dt * (D_Births - D_deaths)

{Auxiliary equations}
D_Births = .15*D_Phi
D_deaths = .00015*D_Phi*D_Phi
```

Parallel C

```
#define XS2 66
#define YS2 66
double D_Phi[XS2][YS2], D_Births[XS2][YS2],
D_deaths[XS2][YS2];

void init_pvars() {
  int ix, iy;
 for(ix=1; ix<=s0; ix++) {
   for(iy=1; iy<=s1; iy++) {
     D_Phi[ix][iy] = 10;
     D_Births[ix][iy] = .15*D_Phi[ix][iy];
     D_Deaths[ix][iy] = .0015*D_Phi[ix][iy]*D_Phi[ix][iy];
         }
 }
}

void link_edges() {
                link_edges_flt(D_Phi);
                link_edges_flt(D_Births);
                link_edges_flt(D_Deaths);
}

void cell_dyn() {
        int ix, iy,;
 for(ix=1; ix<=s0; ix++) {
   for(iy=1; iy<=s1; iy++) {
     D_Births[ix][iy] = .15*D_Phi[ix][iy];
     D_Deaths[ix][iy] = .0015*D_Phi[ix][iy]*D_Phi[ix][iy];
     D_Phi[ix][iy] = D_Phi[ix][iy] + dt*(D_Births[ix][iy]-
D_Deaths[ix][iy]);
    }
 }
}
```

of this program is 10.5 s with two transputers and 19 s with one transputer. With eight transputers, we estimate a running time of about 2.5 s. The same program running on the native MAC IIci 68030 serial processor at 25 MHz under Think C requires 435 s, 41 times that required with two transputers and 174 times that required with eight transputers.

SMP GIS interface

The interface between the GIS generated data and the SMP was designed to be simple and flexible so that many different GIS systems could be accommodated. The SMP reads and writes a simple ASCII based raster array format for all spatial data. Most existing GIS systems will also read and write data in some version of this format (with appropriate header files to tell the system the characteristics of the data). We chose the MAP II Interchange Format as a standard for the header files, but this is easily changed for other systems.

One would thus prepare the spatial data in the GIS system of choice and export it as ASCII raster files. The SMP reads these data files and produces results in the same format that can be read back into the GIS system for display and further analysis.

REACTION-DIFFUSION EXAMPLE APPLICATION

This program illustrates the solution of a spatially distributed partial differential equation (by finite difference methods). The computations involved are qualitatively identical to the computations involved in spatial ecosystem modeling. This simple model is used to evaluate the performance of the transputer network on an interesting class of problems.

The model simulates the two dimensional reaction-diffusion dynamics of a single state variable $\Psi(x,y)$ on a 60 by 60 cell grid. The reaction-diffusion dynamics are described by the equations:

$$\Psi(x,y)' = \Psi(x,y) + \text{rate}*(\Psi(x,y) - \alpha(x,y)) + \lambda*\Psi(x,y)*(1 - \Psi(x,y)),$$

$$\alpha(x,y) = \{\Psi(x+1,y) + \Psi(x-1,y) + \Psi(x,y+1) + \Psi(x,y-1)\}/4,$$

where (x,y) denotes a point in a 60 by 60 cell grid, 'rate' is the diffusion constant, and λ is the reaction rate. In this example we set rate = 0.3 and $\lambda = 0.05$. The initial distribution is a noisy sinusoid in x and constant (plus noise) in y, with three localized 'sinks' distributed randomly. At each sink $\Psi(x,y)$ is constrained to -1 and on the boundaries of the grid $\Psi(x,y)$ is constrained to zero. The spatial dynamics are displayed in Fig. 5. The timing results are discussed above.

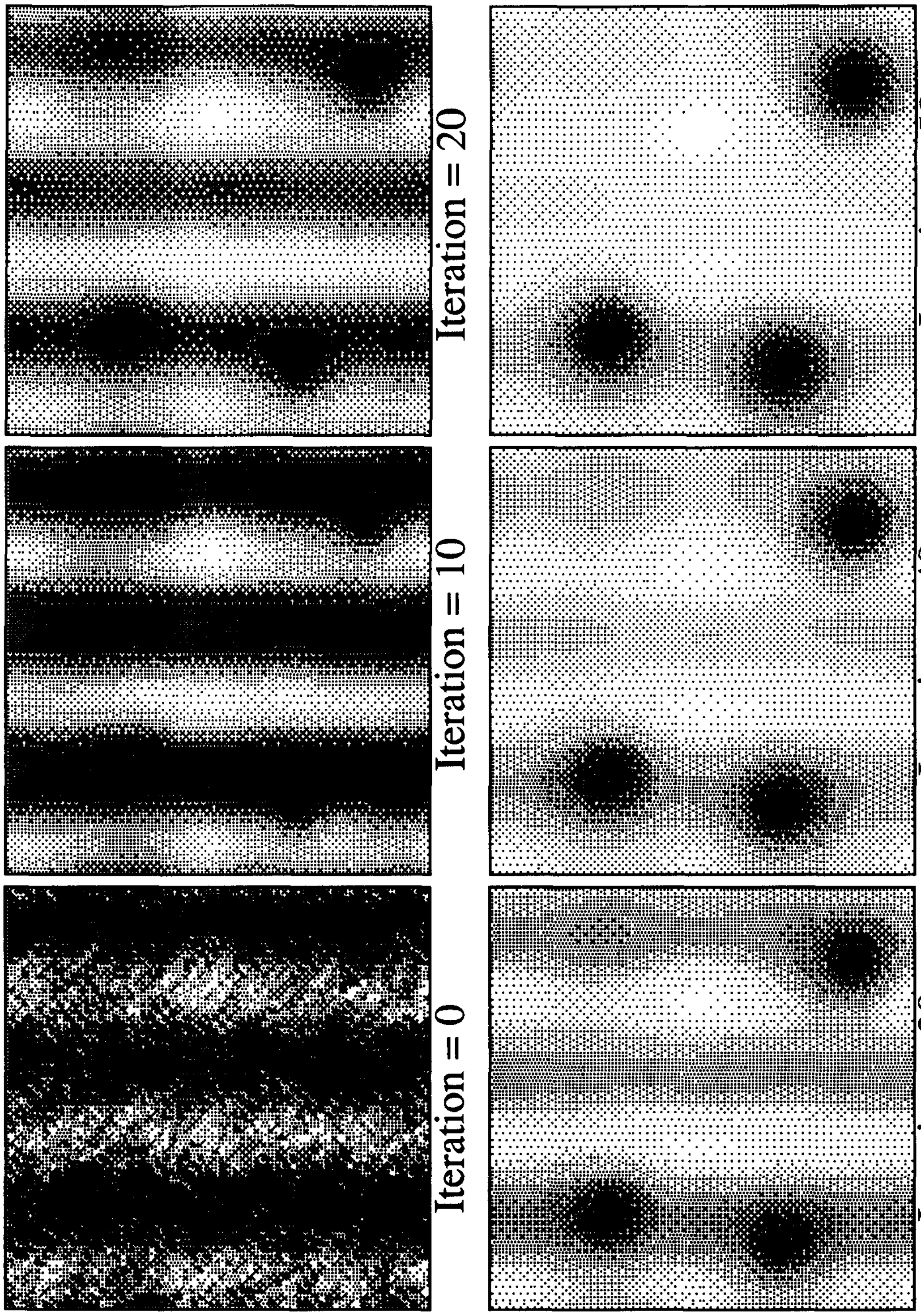


Fig. 5. Sample output from the reaction/diffusion model run on the transputers using a 60 by 60 grid with three randomly spaced sinks. The scale runs from 0 (black) to 1 (white).

As in spatial ecosystem modelling, the dynamics involve only nearest neighbor interactions. The Grid Manager automatically sets up the grid and communication pathways as described above. The array is distributed over the two transputers such that each processor handles the computations for half the grid. The processors must exchange values along the connecting borders of their respective sub-arrays at each time step. We find that in this simple model, the two-transputer system runs approximately 40–50 times faster than the native MAC IIci system. We speculate that this ratio may be larger for more computationally intensive problems for which the computation-to-overhead ratio is larger.

BARATARIA BASIN EXAMPLE APPLICATION

To further test the system described here, we are developing a spatial ecosystem simulation model of the Barataria marsh/estuarine complex in south Louisiana (Fig. 1). Our approach is essentially the same as the one described earlier for the CELSS model application to the nearby Atchafalaya/Terrebonne area. The modelled area is divided into seven habitats: fresh marsh, brackish marsh, salt marsh, swamp, open water, upland, and agriculture. The habitat distributions for years 1956, 1978, and 1988 are shown in Fig. 6. The same basic cell ecosystem model was used for all habitats with the parameters of the model varying as a function of habitat. As in the CELSS model, certain physical conditions, such as water level rise or fall, change in salt or suspended sediment concentration, etc., were capable of driving cellular habitat succession. The buildup of land or the development of open water in a cell depended on the balance between net inputs of sediments and local organic peat deposition on the one hand, and outputs due to erosion and subsidence on the other hand.

As in the CELSS model, the water flux between neighboring cells was calculated as the difference in water levels between the two cells times a flux constant that depends on the habitats of the two cells. All dissolved materials were assumed to be transported by the water flux alone so that the amount of salt leaving a cell was calculated as the concentration of salt in the cell times the water flux out of the cell. This outflux was balanced by influx from neighboring cells calculated in a similar manner. Time series data for external forcing functions such as wind, rain, temperature, pumping stations, tides, etc. were used.

The model is constructed on a 169 by 113 grid of 1 km^2 cells which cover the Barataria basin. Only 5355 (about 1/4) of the 19 097 cell rectangle actually fall within the basin and are considered 'active'; the rest are defined to be 'void'. Within each cell is a local ecosystem model that calculates the state of the system at the next timestep as a function of its current state and inputs

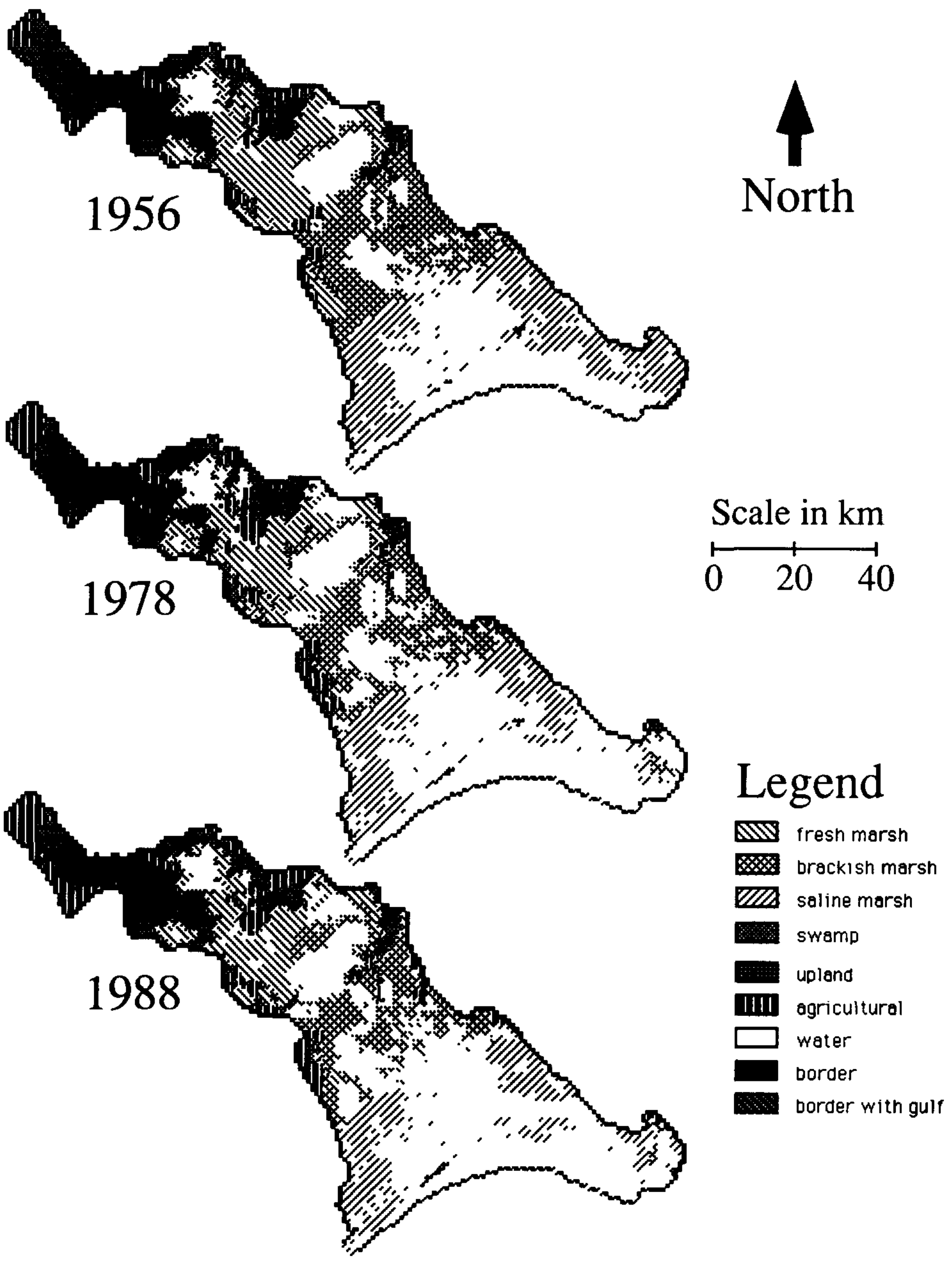


Fig. 6. Historical habitat changes in the Barataria Basin, Louisiana, U.S.A.

(Fig. 2). The cell unit model is essentially the same one used for the CELSS model (see earlier discussion and Costanza et al., 1990).

We used a slightly simplified version of the model shown in Fig. 1 for the timing tests. The model was run over 5355 spatial cells using the system described in this paper on four transputers with 4 Mb of memory each. We did our test runs of the model with no write statements. The model ran at the rate of approximately 3 s/timestep, or 0.56 ms/timestep per cell. The CELSS model (Costanza et al., 1990) ran on a CRAY X/MP at the rate of 0.32 ms/timestep per cell.

Writing large amounts of output data could slow the transputer system considerably. Using eight transputers should double the speed to approximately 0.28 ms/timestep per cell, or slightly faster than the CRAY. However, as noted above, the model that was run on the transputers was slightly simpler than the model that was run on the CRAY, so that the comparisons are slightly biased. But we can conclude that eight transputers will run this type of model at approximately the same speed as the CRAY X/MP.

We are still in the development stages on this model and do not yet have calibration and scenario analysis runs comparable to Fig. 3 and Table 1 from the CELSS model. But the results will no doubt be similar. The differences are the shorter time required to develop and test the model, and the ability to run the model on a small desktop computer at speeds equivalent to those achievable on the CRAY.

CONCLUSIONS

Parallel computer hardware and software are now well developed enough to allow their use in ecosystem modeling. Parallel systems are particularly well-suited to spatial modeling, allowing relatively complex unit models to be executed over a relatively high resolution spatial array at reasonable cost and speed. When combined (on a user-friendly microcomputer like the Macintosh) with dynamic model development software (like STELLA) and GIS software (like MAP II and GRASS), one has a powerful yet easy to use spatial modeling workstation. We have demonstrated these techniques by applying them to a spatial modelling application involving simulations of the Barataria basin in southern Louisiana. These simulations have demonstrated that a Macintosh desktop computer enhanced with relatively inexpensive parallel processing hardware can rival the CRAY supercomputer in speed of computation. In terms of ease of use, the desktop environment is far superior. The spatial modeling workstation we have developed (Fig. 4) should allow the application of advanced computer modeling techniques to a much broader range of problems, especially those involving detailed, spatially articulate ecosystem and landscape modeling.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation, Grant No. BSR-8906269 titled: 'Landscape modeling: the synthesis of ecological processes over large geographic regions and long time scales', R. Costanza and F.H. Sklar, Principal Investigators; and by the U.S. Fish and Wildlife Service under Purchase Order No. 84110-8900682 titled: 'Parallel processor computer modeling system development', R. Costanza, Principal Investigator. We would like to thank E. DeBellevue, S. Tennenbaum and two anonymous reviewers for helpful comments on earlier drafts.

REFERENCES

Boumans, R.M.J. and Sklar, F.H., 1991. A polygon-based spatial model for simulating landscape change. Landsc. Ecol., in press.

Burrough, P.A., 1986. Principles of Geographic Information Systems for Land Resources Assessment. Clarendon Press, Oxford.

Colwell, R.K., 1974. Predictability, constancy, and contingency of periodic phenomena. Ecology, 55: 1148–1153.

Costanza, R., 1987. Simulation modeling on the Macintosh using STELLA. BioScience, 37: 129–132

Costanza, R., 1989. Model goodness of fit: a multiple resolution procedure. Ecol. Modelling, 47: 199–215

Costanza, R. and Sklar, F.H., 1985. Articulation, accuracy, and effectiveness of mathematical models: a review of freshwater wetland applications. Ecol. Modelling, 27: 45–68.

Costanza, R., Sklar, F.H. and Day, J.W. Jr., 1986. Modeling Spatial and Temporal Succession in the Atchafalaya/Terrebonne marsh/estuarine complex in south Louisiana. In: D.A. Wolfe (Editor), Estuarine Variability. Academic Press, New York, pp. 387–404.

Costanza, R., Sklar, F.H. and White, M.L., 1990. Modeling coastal landscape dynamics. BioScience, 40: 91–107.

Forrester, J.W., 1961. Industrial Dynamics. MIT Press, Cambridge, MA, pp. 464.

Gardner, R.H., Milne B.T., Turner, M.G. and O'Neill, R.V., 1987. Neutral models for the analysis of broad-scale landscape pattern. Landsc. Ecol., 1: 19–28.

Kadlec, R.H. and Hammer, D.E., 1988. Modeling nutrient behavior in wetlands. Ecol. Modelling, 40: 37–66.

Kessell, S.R., 1977. Gradient modeling: a new approach to fire modeling and resource management. In: C.A.S. Hall and J.W. Day, Jr. (Editors), Ecosystem Modeling in Theory and Practice. Wiley-Interscience, New York, pp. 576–605.

Krummel, J.R., Gardner, R.H., Sugihara, G. and O'Neill, R.V., 1987. Landscape patterns in a disturbed environment. Oikos, 48: 321–324.

Maguire, L.A. and Porter, J.W., 1977. A spatial model of growth and competition strategies in coral communities. Ecol. Modelling, 3: 249–271.

Mandelbrot, B.B., 1977. Fractals. Form, Chance and Dimension. W.H. Freeman and Co., San Francisco, CA.

Mandelbrot, B.B., 1983. The Fractal Geometry of Nature. W.H. Freeman and Co., San Francisco, CA.

Risser, P.G., Karr, J.R. and Forman, R.T.T., 1984. Landscape ecology: directions and approaches. Special Publication No. 2. Illinois Natural History Survey, Champaign.

Show, I.T., Jr., 1979. Plankton community and physical environment simulation for the Gulf of Mexico region. Proceedings of the 1979 Summer Computer Simulation Conference. Society for Computer Simulation, pp. 432–439.

Sklar, F.H., Costanza, R. and Day, J.W., Jr., 1985. Dynamic spatial simulation modeling of coastal wetland habitat succession. Ecol. Modelling, 29: 261–281.

Sklar, F.H. and Costanza, R., 1991. The development of dynamic spatial models for landscape ecology: a review and prognosis. In: M.G. Turner and R. Gardner (Editors), Quantitative Methods in Landscape Ecology. Springer-Verlag Ecological Studies 82, New York, pp. 239–288.

Turner, M.G., Costanza, R. and Sklar, F.H., 1989. Methods to compare spatial patterns for landscape modeling and analysis. Ecol. Modelling, 48: 1–18.

Urban, D.L., O'Neill, R.V. and Shugart, H.H., 1987. Landscape ecology, a hierarchical perspective. BioScience, 37: 119–127.

Weins, J.A., Crawford, C.S. and Gosz, J.R., 1985. Boundary dynamics: a conceptual framework for studying landscape ecosystems. Oikos, 45: 421–427.