INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Parallelization Issues in Forest Fire Modeling and Related Problems

By
Nashib Qadri

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

May 1997

© Copyright
1997, Nashib Qadri

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-22109-1

Carleton University
Ottawa, Canada K1S 5J7

50 1942-1992

Thesis contains black and white and/or coloured graphs/tables/photographs which when microfilmed may lose their significance. The hardcopy of the thesis is available upon request from Carleton University Library.

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis,

Parallelization Issues in Forest Fire
Modeling and Related Problems

submitted by

Nashib Qadri

Dr. Evangelos Kranakis
(Director, School of Computer Science)

Dr. Jörg-Rüdiger Sack
(Thesis Supervisor)

Carleton University

May 1997

Abstract

The recognition of parallel Geographic Information Systems (GIS) as a significant new area of research, led to the conception of the NEighborhood MOdeling (NEMO) project. The goal of the project was to offer the performance enhancements of parallel computing while abstracting away its complexity. In this thesis, we implement two applications, a fire modeling application and a thinning application to exploit the capabilities of the NEMO system.

The Parallel Fire Behavior Prediction (PFBP) system is based on the Fire Behavior Prediction (FBP) system designed by Forestry Canada. It computes the fire spread using the FBP system and then predicts the time at which the fire will affect neighboring cells. It uses the elliptic fire shape to predict the region affected by the fire. The final fire shape is a composition of many smaller ellipses. The PFBP system offers a significant advantage over the FBP system in that it accurately models the spread over a large area since it integrates maps into the model.

A number of methods to model the spread of the fire to the neighborhood are demonstrated and compared. The output generated by the PFBP may be used to generate fire management data after finding the skeleton of the areas that are not on fire. Several methods of finding skeletons are discussed along with their relative merits and their applications to GIS, such as fire control and map overlays.

These applications were the first, not to be developed by the authors of the tools, to exercise and test the capabilities of the NEMO parallelization tools. Both applications are very useful to fire modeling, an important area in management of forest resources.

We also address the issues of performance gain and usability, when developing real applications in the NEMO framework. As well, empirical evidence collected during the course of this study suggests that characterizing the input images before processing can provide valuable insight into the efficient data distribution and therefore, better performance.

Acknowledgements

I would like to thank, first and foremost, my supervisor Dr. Jörg-Rüdiger Sack, without whose vision, leadership and advice, this would not be possible. I would also like to thank him for encouraging me strongly to spend a year at the École Normale Supérieure in Lyon, France. I would also like to deeply thank my supervisor at École Normale Supérieure, Afonso Ferreira for introducing me to new areas in computing.

I would like to thank Doron Nussbaum for many interesting discussions, providing insight into the vast domain of GIS and some of the seminal ideas in this thesis, Mike Wotton at the Petawawa National Forest Institute. for graciously providing a version of the FBP model, and lots of advice on the correctness of PFBP model. I would also like to thank David Roytenberg, for providing details about the workings of the cellular automata driver and for many discussions. David Hutchinson and Mark Lanthier for their perspective on various communication and tools issues.

John Hogg, was very generous with his advice and time to help me stay on schedule and not lose focus in the critical concluding stages of my thesis. Thanks to Chulaka Ailapperuma, who helped to run some of my tests and Achint Sandhu for proof reading the closing chapters at such short notice.

I would also like to mention Dr. Sack, Dr. Santoro for inspiring me during my undergraduate years to pursue higher studies, and to my parents for being supportive of a higher education from the very early stages of my education. Thanks also to everyone else that supported me in this endeavor.

Contents

Abstract	iii
Acknowledgements	v
1 Introduction	1
1.1 Motivation	2
1.2 Overview	4
2 Parallel Computing	5
2.1 Sequential Computing	6
2.2 Parallel Computing	7
2.2.1 Taxonomy of Parallel Computers	7
2.2.2 Address-Space Organization	8
2.2.3 Computational Concurrency	10
2.2.4 Applications	12
2.3 The Alex AVX Series II Computer	14

3	Geographic Information Systems	15
3.1	GIS Terminology	16
3.2	Data Collection and Storage	18
3.2.1	Data Collection	18
3.2.2	Automating Data Collection	19
3.2.3	Source of Error in GIS Data	20
3.2.4	Data Storage and Exchange	21
3.3	Parallel GIS	21
3.3.1	NEMO	22
3.4	Applications-Forest Fires	24
3.4.1	Fire Danger Monitoring	25
3.4.2	Simple Forest Fire Models	27
3.4.3	Canadian Forest Fire Behavior Prediction (FBP) System	29
3.5	Applications-Skeleton Finding	31
3.5.1	Applications of Skeletons	32
3.5.2	Parallel Thinning Algorithms	33
3.5.3	Errors in Skeletons	34
4	Parallel Fire Modeling	35
4.1	The Parallel Fire Behavior System	37
4.1.1	Inputs and Outputs	39

4.2	Fire Propagation to the Neighborhood	42
4.2.1	Vector Projections	43
4.2.2	Bounding Box - Cell Boundary Intersection	46
4.2.3	Ellipse - Cell Boundary Intersection	47
4.3	PFBP Implementation Issues	49
4.4	Testing	53
5	Skeleton Finding	58
5.1	Exploiting Parallelism in Skeleton Finding	60
5.2	Skeletons of Binary Images	61
5.2.1	Thinning Method	62
5.2.2	Fire Fronts Method	65
5.3	Skeletons of Gray Scale Images	66
5.3.1	Iterated Pixel Removal method	67
5.4	Parallel Skeleton Finding	69
5.4.1	Parallel Implementation	70
5.5	Testing	73
6	Results and Performance Analysis	77
6.1	Performance gains in the PFBP	78
6.1.1	Propagation Driver	79
6.1.2	Analysis	80

6.2	Performance gains in skeleton finding	82
6.2.1	Cellular Automata Driver	83
6.2.2	Analysis	84
7	Conclusion and Future Work	88
7.1	PFBP System	89
7.2	Skeleton Finding System	90
	Bibliography	92

Carleton University
Ottawa, Canada K1S 5B6

Text complete; leaf X omitted in numbering.

List of Figures

3.1	Raster and Vector Representations of an Object	17
3.2	Elliptical fire shape with spread axes	29
3.3	Skeleton of the letter A	32
4.1	A folding sequence showing how a raster is mapped to 4 processors. The raster is shown as a grid of 8×8 blocks in A. B to E show how the raster is folded onto the 4 processors. F to H show the unfolding back to the 8×8 grid. I shows the final mapping of the blocks. (Courtesy of Doron Nussbaum, Paradigm Group, Carleton University	38
4.2	Components of the Parallel Fire Behavior Prediction System	39
4.3	Neighboring cells affected by fire spread	40
4.4	Elliptical fire after translation and rotation	41
4.5	Errors due to simplistic neighborhoods	43
4.6	Projections of head and back fire onto the east-west axis	44
4.7	Bounding box of a fire	47

4.8	Fire progression in the PFBP System. The exterior colors (thin red and yellow bands) correspond to the fire states while the interior bands (including the interior red and yellow bands) correspond to the time range in which the cells caught fire. In (b), one of the processors has not communicated its data to the display driver but it has done so by (c).	57
5.1	Quench points between elliptic fires	59
5.2	Correct (a) and incorrect (b) locations for a skeleton	63
5.3	Propagating fire fronts and quench lines	66
5.4	Incorrect deletion of active border pixels in the propagation driver . .	69
5.5	Skeleton Finding in the NEMO framework	71
5.6	Location of neighbors during skeleton finding	73
5.7	Progression of skeleton finding (thinning) in an image with roads, rivers and lakes. Note: The number of connected components is the same in the original image (a) and the final image (d) after 20 iterations. . . .	76
6.1	PFBP Computation Time [Block size: 32×32 , 6 fires]	79
6.2	Speedups achieved for the PFBP System	81
6.3	Effect of block size on running time [16 procs., 6 fires]	82
6.4	Effect of number of fires on computation time [16 procs., Blocksize: 32×32]	83
6.5	Computation time for uniform and clustered pixel distributions . . .	85
6.6	Computation time for various image sizes	86
6.7	Effect of the number of processors on computation time	87

Chapter 1

Introduction

"At the time of writing (1990) it is possible to buy parallel processing machines... However, a great deal needs to be done to make them generally usable, but when that day arrives, even a desktop computer will be able to outperform many of the large serial machines in use today". (Paul M. Mather, Computer Applications in Geography, 1990, John Wiley and Sons).

Rapid new developments in computing are changing the way problems are solved today. During the infancy of computing, harnessing the power of computers was really difficult since tools were not given much importance. However, the extensive development of new tools that enable technology to be easily used, has resulted in scientific and other disciplines adopting computers as a tool to aid analysis and decision making. Hence, computers are being used extensively in solving scientific problems, modeling natural phenomena, tracking movement of people, goods and services, consumer spending habits etc.

Geographic Information Systems(GIS) are computer based tools used for the collection and analysis of geographic data [88, 69]. GIS are used for resource management, and are required to handle large volumes of data, as discussed in Chapter 3. Since the ability to handle elaborate models and large volumes of data is limited by the computational power available, new ways of providing computational power to such applications need to be explored.

Traditional computers with a single processor are fast achieving the limits of their performance. Improving their performance is very costly and yields low returns in improved performance. Parallel computing is a branch of computing that offers high performance by making multiple processors work in concert. There are several issues that must be addressed when using parallel computers as discussed later in Chapter 2. Since GIS often require computation intensive operations and the ability to process large volumes of data, parallel computing seems like a natural choice to improve GIS performance.

1.1 Motivation

Most existing GIS are based on uniprocessor systems since such machines are inexpensive, do not require advanced skills to operate and are easy to program. In the last decade however, there has been a tremendous amount of research effort directed towards parallel computers, parallel programming and tools. Besides, as parallel computation becomes more mainstream, cheaper and more standardized tools are becoming available. With increasing cooperation between governments and universities, increasing numbers of non-profit government organizations will be allowed access to parallel computing facilities present in research labs at universities. This also allows for a much needed transfer of knowledge from academia to agencies that need to apply the *know how*.

Since it is possible to achieve a high degree of parallelism in raster based GIS as described in Chapter 3, it seems logical that GIS should take advantage of the advances in parallel computing technology and offer much greater capabilities to the user. In order for users to adopt this new technology, the inherent complexity of parallel programming should be hidden from them. The NEighborhood MOdeling system (NEMO) 3.3.1 has been developed by the PARADIGM group at Carleton University, with the goal of providing transparent parallelism. This provides application developers with tools to rapidly develop parallel applications while hiding the parallel complexity behind a parallelism abstraction layer(PAL).

Fire modeling is a common GIS application since fire prediction can save lives and property. There are a variety of tools used to model forest fires, such as cellular automata. One cellular automata based model is the Brinch-Hansen model [54]. The model is simple to allow easy implementation and does not claim to be very realistic. There are other forest fire models that are based on empirical data collected from controlled experiments and wild fires, that are more accurate, but more intensive in terms of computation. One such model developed by the Fire Danger Group at Forestry Canada, is the Fire Behavior Prediction (FBP) system [49], and is detailed in Chapter 3, Section 3.4.3.

Existing models must be sped up so that practitioners can use more elaborate and realistic models that can run in an acceptable time frame. The FBP system is a point based model in that it predicts the spread of a fire from the ignition point. The parallel fire behavior prediction system (PFBP) exploits the parallelism provided by NEMO and extends the FBP system to work with maps so that entire regions can be analyzed using it. Consequently, the PFBP is able to model the fire spread over larger areas more quickly. As a result of the heterogeneity in the input maps, anisotropic patterns are observed during the propagation of the fire. This models real fires much better than the simplistic Brinch-Hansen model.

The skeleton or medial axis of an image is another image that contains only thin lines representing its significant features. It maintains the connectivity and the topological information of the original image. The main uses of skeletons are pattern matching and analysis [74]. The useless information that is contained in the thicker parts of the image is removed so that the matching and analysis phase have to process less information. The skeleton of an image is usually defined and obtained by a distance transform(DT) [90, 18] or by thinning [91]. Skeletons are also used to find the *quench points* of a fire. A quench point is a point on the line where two fire fronts meet, thus extinguishing the fire [100].

1.2 Overview

The focus of this thesis is to study the feasibility of the tools required to use parallel computers to solve GIS problems. It does not attempt to create new models for geographic processes but indicates how existing models may be run on high performance computers thus resulting in more accurate and timely results. Chapter 2 introduces parallel computing, its uses as a tool to obtain higher performance and its various aspects, relevant to this thesis. Chapter 3 provides a high level overview of GIS and relevant issues such as the collection, access and processing of spatial data. It goes on to highlight new developments in GIS from a computing perspective.

The parallel fire behavior prediction (PFBP) system discussed in Chapter 4, takes as inputs, maps representing the fuel type and weather conditions and produces as output, a map of the ignited regions. Chapter 4 also discusses implementation details of the PFBP and alternatives to the current implementation. Chapter 5 takes a look at implementations of some skeleton finding algorithms, analyzes their strengths and weaknesses and then discusses improvements that were made to enhance the performance of these algorithms. The application of skeletons to the collisions of forest fire fronts is examined along with their application to pattern recognition in a GIS context.

Chapter 7 provides insight into the performance gains obtained by running the GIS applications discussed in Chapters 4-5 on the Alex AVX II parallel computer. It also discusses the limitations of current models and possible extensions due to the availability of new parallel tools. Finally, Chapter 7 presents some conclusions and directions for future research.

Chapter 2

Parallel Computing

Problems encountered in scientific computing often require a large amount of computation and storage. The need to solve these complex computational problems has led to a great increase in the demand for high performance computers. Problems in computational fluid dynamics, finite element analysis, molecular dynamics modeling and N-body problems etc. are considered to be too difficult to be solved by sequential computers in a reasonable time frame.

In many applications, visualization of the processes allows scientists to study the processes in great detail, without having to spend the money on devising expensive experiments and collecting new empirical data until they are reasonably sure that they have an effective model [25, 61, 87]. The incredible growth in the computational power available to the end user has seen unprecedented growth in the last ten years. It has made it possible to analyze very difficult problems and increased the appetite for more computational power to solve such problems [62, 80].

2.1 Sequential Computing

Sequential computers today use the von Neumann model of computation. The processor fetches the next instruction and data from memory, and then performs the next instruction, thereby processing the data. Such computers are referred to as *Single Instruction Single Data (SISD)* computers since they have a unique stream for instructions and a unique stream for data. Since all the data must pass through the CPU, the single data channel becomes a limiting factor. This can be overcome by using a technique known as *memory interleaving* where the memory is divided into a number of independently accessible blocks, creating multiple data channels from the memory [35]. This reduces the latency involved with memory access, but the processing is still performed by a single CPU resulting in only one data channel through the CPU.

Another way to speed up the performance is by using *cache memory* which is a small amount of high-speed, low-latency memory present on or very near the CPU and acts as a buffer for the higher latency memory exterior to the CPU. It uses the principle that data that have been accessed once are likely to be accessed again and that data stored in the same region are more likely to be accessed at close points in time. As a result, when a datum is accessed, a certain number of locations before and after it, forming a *block*, are stored in fast memory [35]. The next memory access is very likely to be some data present in the cache memory and can thus be accessed very fast.

The rate of execution of instructions can also be increased by overlapping the execution of an instruction with the operation of fetching the next instruction. This is known as *instruction pipelining* [35]. *Execution pipelining* is a technique that allows instructions in different states of execution to execute simultaneously in different functional units of the CPU [35].

In the past, greater performance could be obtained from uniprocessor based machines by multiple execution pipelines, branch prediction and instruction and data caching. Most of these techniques have been exploited to a very great degree. The

cost of extracting greater performance out of uniprocessor machines is increasing rapidly since the clock speeds and integration of such processors are approaching fundamental physical barriers [35, 43, 60]. Sequential machines often do not exploit the distributed nature of the data itself [43] and are unable to exploit this fact. Hence, new approaches are needed for cost effective performance improvements.

2.2 Parallel Computing

Parallel computing is a branch of computer science that attempts to harness the computational power of multiple independent processors to solve complex computational problems. The basic idea is very simple, i.e., the problem must be broken down into smaller sub-problems and then the solutions to each of the sub-problems are reassembled to find a solution to the original problem. It becomes immediately apparent that there are some issues involved above and beyond just being able to solve the problem, such as the distribution of the data and intermediate results, synchronization to keep the data consistent and re-assembling of the results to find a solution to the original problem. Some of the major issues are discussed in Section 2.2.2 through Section 2.2.3.

2.2.1 Taxonomy of Parallel Computers

In 1966, a seminal paper by M. Flynn [41], described a taxonomy of computer architectures. This taxonomy was based on the number of instruction and data streams in the computer and is listed briefly as follows:

SISD Single Instruction Single Data. This includes typical sequential computers, e.g., most desktop workstations and personal computers.

SIMD Single Instruction Multiple Data. All processors in such a unit synchronously execute the same instruction on different data. Some older parallel computers

but very few modern ones fall into this category, e.g., CM-2, MasPar MP-1 and MP-2.

MIMD Multiple Instruction Multiple Data. Each processor in such a system is capable of executing its own program and most modern parallel computers fall into this category, e.g., nCUBE, iPSC, AVX Series, Cray T3D.

SIMD computers only need one control unit and therefore less memory to store the program. They are naturally suited to *data parallel* programs, i.e., where the same set of instructions is executed on a large data set. MIMD computers require more complex logic at each processor, but are more versatile since each processor can run its own program. MIMD computers often require special hardware for fast synchronization while SIMD computers are inherently synchronized. Since the data are divided amongst multiple processors in both SIMD and MIMD computers, the amount of data that must be processed by each processor is reduced and can often be stored in main memory resulting in a faster turnaround time.

2.2.2 Address-Space Organization

Parallel computers have either a single address space, an address space distributed over the various processors or a distributed address space that gives the illusion of a single address space. Since these are fundamental differences, the address space is also used to categorize parallel computers.

Shared Memory

In a shared memory parallel computer all processors have access to the same memory banks[60]. When a processor wants to communicate data to another processor, it simply writes the data to the shared memory and another processor can then read this data[43, 50]. Shared variables, used for synchronization, may be created using shared memory [43, 50]. The Cray C-90, KSR-1 are shared memory parallel computers.

Shared memory seems to be a natural extension for the sequential model since it provides a single address space similar to the sequential model [43, 60]. In order to provide easily usable shared memory, it is often necessary to give the user a more high level view of the world and the logic for shared memory must be moved to the operating system kernel. This division of labour between the kernel and user program may significantly affect the performance of the system [64].

UMA Uniform Memory Access. In such systems, access to all memory banks, local or remote takes uniform time [35].

NUMA Non-Uniform Memory Access. In such systems, access to local memory banks and remote memory banks may take a different amount of time [35, 92].

Distributed Memory

Each processor, in a distributed memory parallel computer has access only to its local memory and in order to exchange data with another processor, it must send a message to the other processor [35, 43, 60]. In order to be able to send messages, there has to be some routing that occurs for each message. A message may be routed through other nodes before it arrives at its final destination [60]. Synchronization is also achieved by message passing. The AVX Series II and III, the iPSC and nCUBE are distributed memory machines.

The advantage of these machines is that the hardware is simpler to implement but the communication is left to the programmer. In addition, off the shelf elements may be used in the processing nodes instead of specialized hardware.

Distributed-Shared Memory

There are shared memory parallel computers which implement logically shared memory using physically distributed memory, such as the CRAY-T3D. This is known as

distributed-shared memory or *shared virtual memory* [5]. We will however restrict our attention to distributed memory computers that communicate using message passing. A detailed treatment of issues in distributed shared memory may be found in [75] while [40] provides implementation details of such a system.

2.2.3 Computational Concurrency

The primary goal of multiprocessing as a means of increasing computation speed relies on the ability to parallelize computation: to partition, allocate, distribute, and schedule work efficiently amongst the large collection of available system resources, while maintaining high rate of utilization of these resources [68].

In order to find parts of problems or entire problems that are parallelizable, it is imperative to find parts of the computation that are independent, all of which may be performed concurrently. It is possible that some parts may complete execution before others. Once they are complete, they wait for other computations to complete before proceeding, if they are dependent on the other computations.

It is apparent that if the processors spend little time waiting and most of the time executing, we will have an efficient solution. One of the approaches to find such a solution is to overlap useful computation with communication by using threads [9, 36]. Overlapping the two has been used very effectively to solve problems in medical applications such as positron emission tomography (PET) [19, 20, 22]. Thus, when a processor is busy processing, the next set of data are sent to it so that when it finishes processing, it may have the next set of data are ready to be processed. Another approach is to use pipelining on architectures that are inherently pipelined such as systolic arrays [48].

Processor Allocation and Scheduling

On a large parallel machine, it is conceivable that one may want to run several diverse jobs simultaneously [85]. Quite often this means that the machine must be partitioned

into several smaller machines where each of the tasks is run on one partition [63]. This approach has the advantage that processors do not have to perform expensive context switches since the processors in each partition are dedicated to a single task. It also has the disadvantage that processors cannot be shared to the same degree as in a context switching mode.

The computational requirements of various tasks may change dynamically and this may make the initial processor allocations quite inefficient. In order to accommodate the dynamic processor requirements, it might be necessary to reallocate the processors, thus rescheduling the processors [103]. To use resources effectively, and keep response times low, applications are often profiled using various criteria and scheduled accordingly [85]. The overhead involved with detecting the load and scheduling the processors must be minimal in order to keep the solution efficient.

Synchronization

During a computation, a processor may be required to communicate with other processors in order to exchange data or synchronize its activities with those of other processors [35, 43, 60]. Synchronization is often necessary in order to keep the data in a coherent state, so that when data is communicated to neighbors, every processor is in a consistent state in its computation.

Synchronization is required in machines that do not have a global clock as is the case in most modern parallel machines. Each node in a MIMD machine can run its own copy of the program and each processor may work on a different set of data. Different processors may run for different periods of times till the completion of one parallel step of processing. At this point, the processors may be required to synchronize before the next step.

State of the art processors available today operate at extremely high clock rates and can process data very quickly while communications over a copper wire however are still fairly slow. In order to reduce the impact of this delay, a minimum amount

of computation must be performed before any data is to be communicated [97]. The speed of the processor and the speed of communication determine this quantity. Hence in order to efficiently solve a problem, the amount of communication must be minimized while keeping the concurrent computation at a maximum.

In more recently proposed parallel architectures, optical communications are used. This allows for greater bandwidth and the problem in classic networks is reversed, i.e., the communication in such networks is faster than the processing [37, 39]. These networks are not yet completely feasible for implementation, but they change the focus of the performance optimizations. Research in optics has made many of the components required for optically interconnected parallel computers more compact and economically feasible for integration and such computers should be realized within a time frame of ten years.

2.2.4 Applications

Parallel computing represents a major step forward for the scientific community since it is geared towards solving complex problems that require a large amount of computation. Different kinds of parallel computers have been used for solving different kinds of problems. [13, 17, 27, 45, 86, 95] discuss the use of parallel approaches to solving fluid dynamics problems without consideration for the underlying architecture, while others discuss issues that may arise when dealing with specific architectures such as arrays of transputers [2, 55, 94], meshes [79] and hypercubes [52, 71].

One of the most important uses of parallel computing is in tackling the fundamental problems of science through large scale simulations. These problems are known as the *grand challenge* problems. Due to limitations in the speed and memory of computers, some of these problems still cannot be solved. Examples of grand challenge problems are condensed matter physics, convective turbulence and mixing in astrophysics. The memory and computing requirements to accurately model such phenomena are so large, that they cannot be accurately modeled in the near term.

Molecular dynamics has been looked at from an architecture independent view by [11, 73, 84, 89] while [8, 10, 70, 78] study the same problem on rings and arrays of transputers. The availability of powerful parallel computers has enabled scientists to study the interactions between complex problems such as fluid dynamics and molecular modeling [102].

In addition to these problems, finite element analysis and its applications to structural analysis have been extensively modeled on parallel computers by [4, 15, 29, 44, 46]. Others, such as [14, 16, 38] discuss these problems as implemented on transputer based parallel computers or other special purpose architectures [6]. In addition to the traditional scientific computations, parallel computing is increasingly being used in other areas such as geographic information systems (GIS) [21]. It is of great interest for computer scientists to explore these new areas since they offer formidable challenges to the current technology.

The emergence of parallel computing is one of the most promising alternatives from a performance perspective. It is not however a panacea that solves all performance problems. There are a number of complex issues that must be dealt with when finding efficient parallel solutions as discussed in Section 2.2.2 through Section 2.2.3. This makes the analysis of the performance of parallel algorithms very architecture dependent.

With the advent of parallel computing however, it is possible to study problems in much greater detail, e.g., it is possible to use very refined meshes during finite element analysis thus modeling the physical processes in great detail. Design parameters can be modified and the simulation rerun to understand their effect on the system. This allows the costs associated with the concrete prototypes to be avoided while using software prototypes, and has led to new levels of productivity in the engineering environment.

2.3 The Alex AVX Series II Computer

The applications implemented in this thesis were implemented on the Alex AVX Series II parallel computer built by Alex Informatique. The largest of these computers has 64 nodes and some additional nodes for providing external access. A node in this computer has a compute engine (an Intel *i860* processor) and a communications processor (an Inmos *T805* processor). The compute engine has 64 Megabytes of main memory while the communications processor has 16 megabytes of main memory.

The AVX Series II computers have a reconfigurable topology. It can be configured as a hypercube, mesh or tree. Each machine has a Sparc 20 as a host processor. The host processor can communicate to the nodes of the machine using the communications provided by the Trollius operating system [33]. The Trollius system contains a multi-layered communications system that is highly configurable. One of the limitations of the compute engine in the AVX processor is that *i860* can only handle one task at a time.

At the lowest level of communication sits the data link layer. The network layer is the next layer of abstraction, while the transport layer provides a high level connection oriented messaging service. The transport layer provides facilities for non-blocking and blocking messaging. Each layer provides another level of abstraction over the underlying layer. Thus, Trollius provides a very versatile and efficient set of communications primitives.

Chapter 3

Geographic Information Systems

The tremendous impact of mankind's existence on the environment is being felt more than ever today due to the explosive growth of the human population base and the pressure it puts on the other plant and animal species. The planet must be able to cope with these pressures for the populations to survive. The first step in being able to cope with these pressures is being able to understand their impact on the environment. This means that there must be tools available to constantly monitor and record the state of the environment, find changes and predict the impact if these changes continued unchecked.

With the advent of advanced technologies into most scientific, business and personal domains, it seems natural to harness these technologies to monitor the environment. In the geographic context, these new technologies offer great potential to automate repetitive tasks, tackle more complex problems decrease the reaction time for response to crises.

3.1 GIS Terminology

A *geographic information system* is a facility for preparing, presenting and interpreting facts that pertain to the surface of the earth [88]. With the advent of computers into all domains, the modern geographic information system is considered to be a configuration of hardware and software that is specifically designed to collect, process and display geographic data.

Geographic information systems are changing the way modern geographic data are handled. They are indeed making it easier to organize, process and store data. GIS generally use two types of encodings for the data, illustrated in Figure 3.1, and described as follows:

Raster: Raster encoding gets its name due to the similarity between the cartographic grid and the pattern of dots on a video image. In such an encoding, there is one value that corresponds to each attribute of the cell. Since each cell represents a fixed area, the amount of storage required depends on the area being covered. If there are $m \times n$ cells in the raster, and k attributes must be stored for each cell, then the total number of values that must be stored is $m \times n \times k$.

Vector: Vector encoding only needs to store the boundary lines of the particular characteristics being stored in a particular layer. The amount of storage required is a function of the number of objects being represented and not the area covered by the map. Hence, vector encoding usually requires far less space than raster encoding.

The following terms are used to describe some of the basic notions about GIS data as they pertain to GIS [47]:

Maps: Maps contain references to *layers* that hold basic data, as well as information about representation and organization of data.

Figure 3.1: Raster and Vector Representations of an Object

Layers: Layers contain information about specific features of the landscape such as topography, vegetation cover, roads etc. A map may overlay multiple layers to produce a visual representation of the region.

Pixel Layers: They consist of a value for each cell in the region, e.g., a layer representing the fuel cover would contain a value corresponding to the type of vegetation present in the area represented by the pixel. Pixel layers may have *legends* associated to them.

Object Layers: Object layers consist of objects, e.g., roads, houses and information for each object. An object layer would contain the specific information relevant to an object such as the name, type etc., a starting point and a vector representing its path across the landscape. Object layers too can have *legends* associated with them which may contain visual representations for the different objects contained in it.

Function Layers: Function layers consist of equations or tables defining how the

values of particular variables depend on the location, e.g., the weight of the fuel in a cell may be a function of the longitude and latitude of the cell.

Legends: Legends are tables which contain visual or other representations for the data contained within the pixel or object layers.

GIS data are often referred to as *spatial data*. Data stored within a GIS may also be time variant and are referred to as *temporal data*. We use the term *geographic data* to refer to spatial and temporal data. In the PFBP system, we use spatial data that is time invariant. This data is presented to the PFBP system in the raster format. There are several maps, one for the fuel cover, the elevation which are input maps. There are two other maps that are computed by applying some functions to the weather and input maps. These are the buildup index and the fine fuel moisture code maps. There are no object layer maps since the PFBP system is based entirely on raster based maps. There are no formal function layers and legends are implied by particular applications.

3.2 Data Collection and Storage

In addition to tools for processing, analyzing and visualizing data, a GIS must provide adequate tools for the creation, collection and storage of vast volumes of geographic data. The tools must be easily usable, reliable and secure so as to allow multiple users to access information simultaneously.

3.2.1 Data Collection

GIS data are collected by various means such as automated mapping, remote sensing and various surveying techniques. A significant portion of these data are in different formats since they are generally tailored to specific application areas such as land and resource management, transportation and health care etc. [59] One of the most

important issues facing GIS users and creators is the efficient handling of large volumes and different formats of the data.

Remotely sensed data can only be used as spatial data after some preprocessing of the data. This requires linkages between the GIS and the Image Analysis System (IAS). Remotely sensed data are available at a variety of resolutions ranging from one kilometer to ten meters and have a wide range of uses, e.g., general continental land use, crop condition at a regional level etc. GIS and IAS are strongly coupled in both directions since GIS can be used for further analysis of the remotely sensed data [12].

3.2.2 Automating Data Collection

Vector digitizing is a commonly used means of creating vector data and consists of recording coordinate and topological information from a source diagram such as a vector map. The availability of the relatively low cost global positioning system (GPS) makes the re-surveying of topological information with direct digital data input a practical and cost effective option for many organizations [69]. Very often, the process of converting existing maps into digital information is a tedious manual process and is error prone due to the sensitivity of the accuracy of the conversion devices and the operator. The stretching of sheets and overlaying of several sheets to form a single map can introduce errors into the digital data as well. For a more detailed treatment of errors in GIS data, refer to Section 3.2.3.

Another technique often utilized is the collection of digital terrain data after the processing of remote sensing data [56, 57]. Conventional terrain feature detection and classification methods such as maximum likelihood classifier classify an observed signal value such as multi-spectral data, X , to the most plausible class, W_i . Little spatial information is applied to identify terrain features except cell wise texture information. Spatial interpretation is usually applied after classification. Since conventional image analysis systems cannot handle spatial information such as shape and geometry, it is invaluable to accommodate spatial information processing functions for automated terrain feature detection [56].

3.2.3 Source of Error in GIS Data

One of the most important considerations that must be taken into account, when analyzing GIS data, is that the data may not be error free. It is impossible to have completely accurate data due to the nature and volume of data being monitored. Hence, the error in the data must be within an acceptable range. There are several reasons that there may be errors introduced into the data. Some of them are outlined below:

Type of data: Stored in data structures, geographic information is always discretized and has sharp boundaries. This is not usually the case in reality. The data are gradual and fuzzy since vegetation types do not have clearly delineated boundaries and generally changed to a different type very slowly.

Device Sensitivity: Errors are often introduced since the instruments used for measurement have limited sensitivity. The scale at which these measurements are being made, the sampling frequency etc. also introduce potential errors into the data.

Data Models: When using vector representation, some degree of error is introduced since an edge describes a level of detail, that is not discernible in the real world. Similarly, in raster data, the averaging or aggregation of conditions by imaginary cells is also susceptible to error. Processing the data and transformation between different formats may introduce error into the process as well.

Thus, when using the collected geographic data in predictive models as the PFBP system, it is very important to understand, that after a model reaches a certain degree of accuracy, predictions cannot be made more accurate by enhancing the model since the error introduced by the data bounds the accuracy of the predictions. If however, the accuracy of the data is increased, then enhancing the model will yield more accurate predictions.

3.2.4 Data Storage and Exchange

Once the spatial data has been obtained using one of the methods mentioned in 3.2.1, it must be stored in a format that makes it easy to retrieve large volumes of data easily and to perform complex spatial queries.

Since it is very expensive to collect data, data are often exchanged by cooperating organizations so that they need to be collected just once. Similarly, different arms of the same organization, that are located in distinct geographic locations may need to communicate spatial data to each other. As a result, there is a need to establish protocols to store the data and exchange it. Due to the large volumes of data that are often involved, data compression is a major issue as well. Several well know schemes such as quadtrees, octrees etc. exist for compression.

Concurrency Control

GIS must allow for concurrent access of data while maintaining data integrity, since they may have multiple users. The GIS must also allow several users to access the same data at the same time for read-only operations, but it must restrict multiple users from accessing the same data for write operations. It must maintain a high degree of concurrency as well [58]. Hence, it may allow different users access to different layers within the same map and may also allow different users access to different tiles within the same layer. The latter requires much more complicated schemes to implement it and therefore involves some amount of overhead as well.

3.3 Parallel GIS

Some GIS operations are very expensive in terms of both, computation and I/O. On sequential machines, computations often take hours to complete. This reduces the ability to vary the parameters and recompute the results for a new set of parameters.

Hence, it limits the user from studying the behavior of the system under varied conditions. As discussed in Chapter 2, parallel computing is an attractive option that offers high performance computing. Thus, it seems to be a natural choice to use parallel computing as a basis for the next generation of high performance GIS. However, the first step in using parallel computing is finding a mapping between GIS problems and parallel computing techniques that solve them [67].

Parallel computers are hard to program and difficult to use due to a lack of tools such as debuggers, communication libraries etc. Even if the tools are available to program them, it is not easy to write efficient programs that obtain optimal or near-optimal speedup without the overhead of developing efficient parallel algorithms. This problem arises from the very fundamental issue of mapping the program architecture to the physical architecture. Most users that are experts in their domain should not be expected to become experts in parallel computing before using it to solve their problems. Thus, there is a need to offer tools that hide the inherent complexity from the end user, but offer them the power of parallel computing.

In raster based GIS, a high degree of parallelism may be achieved since the calculations required to process a cell often depend solely on its value and that of its immediate neighbors. Even in vector based GIS, it is possible to use parallelism since independent objects may be processed concurrently. This however requires more sophisticated techniques than in raster GIS. The data-parallel model of parallel computation is one that is intuitive and applies very well to the GIS domain. Also, this model is more intuitive than the sequential model since it captures the locality and independence of different regions. It is suitable for problems that deal with regular spatial and temporal structures and thus seems to be suited to raster based map analysis.

3.3.1 NEMO

NEMO is a parallel system designed for neighborhood modeling of raster data. The aim of NEMO is not to replace, but rather to complement existing sequential GIS

systems in tasks which are very expensive in terms of computation time. Thus, NEMO provides a very fast computational engine for a gamut of sophisticated problems based on raster data [32].

As mentioned in Section 3.3, it should not be required of users in the GIS domain to understand parallel hardware or the complicated programming paradigms of parallel computers. Hence, it is the goal of NEMO to provide *transparent parallelism*. This means that the user must be able to develop neighborhood models without being aware of the underlying parallel issues. NEMO provides the ability to perform neighborhood modeling using three widely accepted mechanisms:

1. Cellular Automata
2. Propagation Modeling
3. Neighborhood Modeling

A *neighborhood* is a set of locations defined as using cartographic distances and/or directions from a given cell. The cell itself is known as the *focus*. In a typical GIS application, neighborhood modeling is used to model the occurrence of various phenomena, e.g., if a cell has a neighbor that is on fire in a forest fire model, it is more likely that in some finite time interval t , this cell will catch fire than a cell that has no burning neighbors. Thus, the use of neighborhood modeling is very obvious in a variety of situations such as spreads of pests or fires, computation of earthquake prone zones etc.

The previous example demonstrates clearly that only the cells that are on fire will affect their neighbors while the rest need not compute their effect on the neighbors. The propagation driver takes this into account and processes the cells on a priority basis. Hence, the cells that are burning the 'fastest' affect their neighbors quicker than other cells. Thus, the propagation driver allows for the modeling of much more realistic sophisticated models and is also more efficient since it keeps track of an active border of spread. The propagation driver was used in order to model the forest fires

in the PFBP system in Chapter 4 while the cellular automata driver was used for computation of the skeletons in Chapter 5.

NEMO also has two other components, the raster data manager (RDM) and the display manager (DM). These are the server components of NEMO and provide services to the basic drivers as well as applications that are built on top of NEMO. The RDM provides access to multiple rasters concurrently and supports sophisticated operations such as logical tiling, tile locking, data compression and transparent data conversion. The DM provides access to display facilities for applications as well as the neighborhood modeling drivers. It allows the spatial data to be displayed and supports zooming and animation of the results [32].

3.4 Applications-Forest Fires

GIS have numerous applications in the management and monitoring of natural phenomena. They have been used for a variety of applications such as monitoring the habitat of wildlife, watershed management, analysis of land for agricultural suitability, tracking and predicting hazardous phenomena such as earthquakes and forest fires, and oil spills.

Wildfires are fires caused by natural or man-made causes that destroy the vegetation cover and are difficult if not impossible to control. In 1993, which was one of the worst years in recorded Canadian forest fire history, the damage was estimated to be a billion dollars [76]. In order to reduce the amount of damage caused to life and property, effective means of predicting and managing fire risk are needed. Wildfires are usually caused by lightning, arson, accident or by controlled fires going out of control. Controlled fires are used extensively to manage tree growth and reduce the risk of wildfires.

Wildfires have been studied extensively from the perspective of thermodynamics, chemistry and modeling of the parameters associated with fire prediction and control.

However, their spatial behavior and the use of remote sensing to manage them has not been given much attention until recently. There are several advantages to performing large scale fire wildfire experiments as simulations [24, 47], namely, being able to test various fire control techniques. Instrumentation of fires to measure their physical properties is expensive, difficult and dangerous [24].

Combustion of the forest fuels occurs in three phases:

Ignition: Ignition sources are many as mentioned earlier, and have spatial distributions that are well characterized. It is generally affected by the fuel content of the moisture and the air.

Propagation: It is characterized by pyrolysis. Pyrolysis is the consumption of the fuel and involves the release of gases which cause strong winds thus fanning the fire.

Extinction: During the propagation, the time elapsed before combustion of new fuels is critical and depends on the pre-heating of the fuels. Once, the fire intensity falls too low to cause flaming combustion, the fire burns the fuel that has already caught fire without burning further.

These phenomena are usually indistinct spatially but distinct temporally and a more detailed treatment of these processes may be found in [24].

3.4.1 Fire Danger Monitoring

Grassland fires pose a serious threat to property and livestock forage in the great Northern Plains. Fire danger warnings are broadcast through the radio and television to warn the public about fire danger. The fire danger index is computed by taking into account daily weather conditions, humidity, wind speed, cloud cover and percent green cover (PGC) of the vegetation which represents the fuel condition. Weather condition information is obtained from the National Weather service meteorological

stations within a state. The state forestry agencies provide estimates of the PGC from sample sites that are obtained by measuring the percentage of dry and green components of the vegetation [34].

The accuracy of the PGC is restricted by the fact that it cannot be sampled over very small widespread areas and cannot be sampled frequently due to the human cost of labour involved. In order to overcome this shortcoming in measuring the PGC, Advanced Very High Resolution Radiometer (AVHRR) data was used to monitor the fuel conditions on a statewide basis. Due to cloud cover, it was not always feasible to expect the satellite to provide the required data in pass over the state. A number of passes were used that overlayed the data to create a single clear comprehensive cover of the area in question. The use of this new automated method resulted in a much more accurate calculation of the fire danger index compared to the prior method of sampling selected areas manually [34].

The Canadian Fire Weather Index System, is similarly based on a number of years of research and provides an index computed from the moisture content of three types of forest fuel types plus the effect of the wind on fire behavior [93]. The FWI comprises six components. The first three are moisture codes that follow daily changes in the moisture contents of three classes of forest fuel with different drying rates.

Fine Fuel Moisture Code (FFMC): For litter and other cured fine fuels (dry weight 0.25 kg/m^2).

Duff Moisture Code (DMC): For loosely compacted decomposing organic matter (dry weight 5 kg/m^2).

Drought Code (DC): For compact matter (dry weight 25 kg/m^2).

These codes plus the wind are combined to form two intermediate indices which are then recombined to form the final index, i.e., the FWI [93].

Initial Spread Index (ISI): A combination of wind and FFMC representing the rate of spread without variable fuel quantities.

Build Up Index (BUI): A combination of DMC and DC representing the total fuel available to the fire.

Fire Weather Index(FWI): Computed from the ISI and BUI. Represents the intensity of the spreading fire as energy output rate per unit length of fire front.

3.4.2 Simple Forest Fire Models

There are several techniques used to model forest fires in order to study their behavior. Most models simplify the models to make them computationally feasible. Thus, different models may have varying degrees of accuracy and computational cost. Usually, the simpler the model, the easier it is to implement, but the greater the inaccuracy. The models can be divided into cellular automata based models and propagation based models. Models in both categories are able to model reality quite well depending on sophisticated the spread and weather modeling are.

Cellular Automata Based Fire Models

Forest fire models can be modeled using stochastic cellular automata which is defined on a d -dimensional hypercubic lattice with L^d sites [23, 28]. Each site is occupied by a tree, a burning tree or it is empty. During one time step, all the sites are updated in parallel using the following rules:

1. burning tree $\rightarrow$ empty site
2. tree $\rightarrow$ burning tree (if at least one neighbor is burning).
3. tree $\rightarrow$ burning tree with probability f (if no neighbors are burning)
4. empty site $\rightarrow$ tree with probability p .

The model can be made more sophisticated by giving the trees some degree of immunity, g , against fire. The above model is not realistic from a forest fire modeling

perspective, but is presented to illustrate the basic structure of a cellular automata based model. In [23], the authors demonstrate that the critical behavior of this model and that it behaves like a percolation process in higher dimensions. Brinch Hansen and others have also provided simple models based on this technique [54]. Other cellular automata based models are much closer to reality and use much more accurate rules than the ones cited above to model fires. e.g., the model in [24].

Propagation Based Fire Models

Most propagation based models are derived from closely modeling the physical processes of heating and taking into account phenomena such as the curing of fuels, weather, moisture content of fuels etc. The models are usually based on a combination of accurate physical models that are retro fitted to empirical data collected from experimental and wildfires.

One such model, proposed in [1], discusses the use of Huygen's principle and its advantages in modeling forest fires relative to some other methods. Huygen's principle states that each point on a wavefront may be treated as a source emitting a secondary wave. Thus, each point that is on fire on the fire front is considered to be a point source of fire. It is much faster and more accurate since the processing is limited only to the active border and thus the rest of the cells need not be processed. There are other models such as the FBP system [49], discussed in detail in Section 3.4.3 that are very accurate since they use retro fitted physical models. A fire model for prescribed fire management based on similar techniques is described in [96]. The comparison of the field measurements and the models described in [96, 49] demonstrates the accuracy and realism of these models.

3.4.3 Canadian Forest Fire Behavior Prediction (FBP) System

The Canadian Forest Fire Behavior Prediction (FBP) System is one of the components of Canadian Forest Fire Danger Rating System. The FBP system provides quantitative estimates of the rate of spread of the fire in all directions, the fuel consumption and intensity of the fire. It uses an elliptical growth model to estimate the area, perimeter and behavior of the fire in the flank and back directions.

Figure 3.2: Elliptical fire shape with spread axes

It covers sixteen distinct fuel types, covering most boreal forest fuel types in Canada. A database of over 400 experimental, wild and prescribed fires was used to derive the fire behavior models for the spread rate and fuel consumption. It also provides comprehensive models for crown fires and fire growth.

Derivation of Models

The equations modeling spread and behavior of the fire were developed using field observations and documentation of readily measured variables on experimental fires

in forest stands and clear cut logging slash and then analyzing the data using simple mathematical models and correlation techniques.

In addition to experimental fires, well documented prescribed fires and wild fires were used. Wild fires are very useful at the extreme end of the fire behavior scale, where experimental fires are difficult to schedule and manage. The framework for the analysis of field data are obtained from laboratory based fire research in moisture physics and heat transfer theory.

Inputs and Outputs

The main inputs to the FBP system are the fuel type, weather, topography and geographic location. The fuel type inputs include the vegetation type, the amount of grass cover, dead fir etc. The weather inputs include the wind speed and direction, and other indices that are computed from weather observations such as the build up index and the fine fuel moisture code. The topographical inputs include elevation, latitude and longitude etc. These are detailed in Section 3.4.1. It predicts several characteristics of the fire and its spread pattern such as:

1. Spread rate; the spread rate and distance are measured in the four directions along the major and minor axes of the ellipse.
2. Fuel consumption: measures the weight burned per unit area per unit time.
3. Fire intensity; measures the heat output per unit area and onset of crowning.

Assumptions

There were some simplifying assumptions made in the model as are necessary to make any such sophisticated model tractable. They simplify the model a great deal, but do not significantly affect the accuracy of the model. Some assumptions that are the most relevant to the PFBP system are listed below:

- The equilibrium rate of spread predicted is always the worst case or line source fire equivalent, i.e., the fire is considered to be advancing as a line representing the fire front and not a point.
- To predict the fire intensity, a standard value of 18000 kJ/kg is used for the heat of combustion.
- The back fire rate of spread is not predicted very accurately but, this does not significantly affect the accuracy of the model in the presence of any realistic wind speeds.
- The transition of the fire from a circular fire from a point source to an elliptic fire during equilibrium due to a line source is not very accurate. This does not affect the final results.

As described in Section 3.4.2, this model is very accurate and was thus chosen as the model to be parallelized. It was hoped that the PFBP system would offer realistic modeling capabilities due to increased computational power and extend the realism of the FBP system.

3.5 Applications-Skeleton Finding

The *medial axis skeleton* is a thin line graph that preserves the topology of the region as described in [18], and shown in Figure 3.3. The skeleton is a very useful representation since it may be used to describe the shape, region information and for object recognition. Skeleton finding is also done by *thinning* since the skeleton is often found by deleting pixels on the outside of the image till the only pixels that are left form thin lines that maintain the connectivity of the original image [53].

Shape analysis techniques can broadly be divided into two categories [26], described briefly as follows:

Figure 3.3: Skeleton of the letter A

Boundary Centered Techniques: These techniques analyze shapes by computing features based on the bounding contours. These contours are found by computing a set of critical points that correspond to perceptually significant points.

Region Centered Techniques: These techniques analyze shapes based on their internal structure. The most common example of this is the medial axis transform (MAT).

However, in [26], an application that uses techniques from both categories was demonstrated. The application in [26], was developed in order to better understand the role of the pseudopod in leukocyte locomotion and hence it was necessary to use pattern recognition to analyze these images.

3.5.1 Applications of Skeletons

Skeletons are very useful since they find an image representation that requires significantly less information than the original image. Since the representation has certain constraints, this ensures that the skeletons are in fact, truly representative of the features of the original image. Hence, operations such as the comparison of features,

can be performed in a much shorter time period by comparing the skeletons' features instead of comparing the original images.

Skeletons are used in a variety of domains, the foremost among them being pattern recognition and several applications that make use of pattern recognition. It has been used for the modeling of objects, industrial parts and molecular structures [90]. As well, it has been used for image understanding and character recognition [31, 53, 91]. Skeletons have been used in biological applications such as tracking the locomotion of leukocytes [26]. Skeletons have also been used to find the *quench points* for colliding fire fronts. A quench point is a point where the fire extinguishes itself since two fire fronts meet at this point and all the fuel in the path from the source to the quench point has been consumed. A more in depth examination of quench points and their relationships to skeletons and forest fires is found in Chapter 5.

3.5.2 Parallel Thinning Algorithms

Sequential approaches to thinning have existed for a long period of time, since Rutowitz's seminal paper [82]. But with the advent of parallel computing into many other scientific domains, there has been an increasing interest in parallel algorithms for skeleton finding as well. A fully parallel thinning algorithm is one that applies the same thinning operator to the entire image in each iteration [51].

The amount of communication required to solve this problem on a parallel computer is directly proportional to the size of the neighborhood. Hence, the smaller the neighborhood, the lower the communication overhead. This suggests that 3×3 neighborhoods would be well suited to a parallel solution. However, when deleting a pixel in a fully parallel implementation of a thinning algorithm, a 3×3 neighborhood does not always supply adequate information. Thus, one way to provide adequate information is to use larger neighborhoods. Another way to work around this problem is to divide the algorithm into sub iterations thus serializing the algorithm partially. This is discussed in greater detail in Chapter 5.

There are several papers that discuss the comparative performance of various thinning algorithms and set the criteria for the comparison and absolute performance of such algorithms, most notably [53, 51]. The primary criteria are the neighborhood size and the number of iterations. Some measure of absolute performance is necessary and thus [51] presents some techniques to measure whether or not the algorithm is optimal or not.

3.5.3 Errors in Skeletons

Although skeleton finding has been studied extensively, most skeleton finding algorithms degrade the resulting skeleton very quickly in the presence of noise or non elongated objects. Also, most of the research has focused on finding the skeletons in binary images. However, increasing effort is being directed towards finding the skeleton of grey scale images [3, 83]. Most real images such as chromosomes, machine parts and handwritten characters do not have sharp well defined boundaries but noisy and speckled boundaries [83]. As well, not all images that need to be processed are elongated. Thinning does not keep information about the convex nature of the image.

There exist several techniques to reduce the effect of the above factors on the accuracy of the skeleton, e.g., filters may be applied to the image in order to reduce noise on the boundaries. Usually, application of such filters does not affect the accuracy of the skeleton too much. One way to find the skeleton of grey-scale images is to transform them into binary images and then apply algorithms developed for binary images. This can generate noise in the original image and hence yield an inaccurate result. There have been new advances to directly find the skeleton of grey-scale images as discussed in more detail in Chapter 4. Techniques using a combination of the two techniques described in Section 3.5 are used in order to reduce the inaccuracy of the skeletons of convex shapes.

Chapter 4

Parallel Fire Modeling

Some fire models can be extremely sophisticated, as discussed in Section 3.4, since they use retrofitted physical models. If these models are used, the computation time associated with each location on fire is high. It is often of interest when managing fires to predict the spread over an extensive region rather than a single location. On a raster map, this implies computation of the fire spread for each cell that is on fire. Since non-neighboring locations never affect each other directly, each one of the fires may be processed independently of all other fires and more specifically, each cell that is on fire may be processed independently of all its non-neighboring cells. This provides an opportunity to exploit parallelism when computing the fire spread for independent cells. The NEMO system, described in Section 3.3.1, is a system that provides easy to use parallelism for problems of this nature.

There are several models that use the power of parallel computing to model forest fires using cellular automata [54, 24]. Such models iterate through each cell during one time step and determine the state of the cell for the next time step. Parallel cellular automata based fire models have the following properties:

- quantization of time using discrete time steps
- synchronization at the end of each time step

These two properties make such models simple to implement and debug since the models proceed in discrete time steps. Hence, cellular automata are very popular and are the basis for theoretical and practical models [54, 24].

Even though cellular automata based models are able to model fires, they have some drawbacks. They impose a discontinuity in terms of time. i.e., if iteration one shows the state of the fire after ten minutes, then iteration two will show the state of the fires as at twenty minutes. There is no accurate way of determining the state at fifteen minutes other than choosing a different time step. Interpolation may be used to determine the state at some intermediate time, but it is not always accurate and if it is, then the associated cost of computation is too high. Making the time steps smaller by a factor of k , has the effect of increasing the computation time by a factor of k . Hence, it is not always feasible to use very small time steps.

Fires are typically perceived as propagation or spread based phenomena since an area on fire typically ignites the area around it. Propagation models model spread phenomena much better than cellular automata models since they are a natural match to such phenomena. Hence, fire behavior can be modeled using the propagation driver of NEMO and it was decided to explore parallel implementations of the fire behavior prediction (FBP) system using NEMO.

Propagation models work on the principle that a source cell affects its neighbors at a certain time in the future. Hence, they predict the time at which the neighbors of the source cell will reach a given state. This is in contrast with cellular automata based models which predict the state of the cells at a given time. Different neighbors may be affected at different times. There exists a concept of continuous time instead of discrete time steps and the next cell to be processed is the one which is affected earliest. A cell does not have to wait for synchronization to other cells before proceeding to affect its neighbors. Hence, two non-neighboring cells can easily be processed simultaneously, providing an excellent opportunity to exploit parallelism.

One of the greatest advantages offered by propagation based models is that they are much more efficient than their cellular automata based counterparts. In a cellular

automata based model, the whole raster is processed during each iteration. In a propagation based model, on the other hand, a list of *active* cells is maintained. The active cells are the only cells in which events of interest occur. The rest of the cells need not be processed and for very large problems need not even be brought into memory. This allows the propagation driver to process a small active area at a finer resolution than a cellular automata model. The cellular automata model must work at a lower resolution since every pixel must be processed. The propagation driver of the NEMO system implements these sophisticated capabilities required for efficient propagation models and was thus chosen to be the underlying tool for the parallel fire behavior prediction system.

In order to balance the load fairly, the propagation driver distributes the data using a 2D folding scheme, instead of simply partitioning the raster into as many partitions as there are processors. This allows the clustered data to be redistributed among the processors. Due to the folding, the distance that messages must travel is greatly reduced since it ensures that neighboring partitions of the raster are on neighboring processors.

4.1 The Parallel Fire Behavior System

The FBP system is a point based model, i.e., it makes predictions about the rate of spread and the behavior of the fire in a small geographic area, such as the grey cell on the map depicted in Figure 4.2. It is not devised to work with maps or large areas. It takes various inputs, such as weather, topographical and fuel data. However, it can only process fire spread at a single location, during one run of the program and thus processes only one value of each input parameter. To apply the FBP over a region, the program must be run over and over again treating each point independently of every other point. Thus, it is unable to take into account the effect of the fire on neighboring cells.

In order to accurately predict fire behavior over large areas, it is imperative to

Figure 4.1: A folding sequence showing how a raster is mapped to 4 processors. The raster is shown as a grid of 8×8 blocks in A. B to E show how the raster is folded onto the 4 processors. F to H show the unfolding back to the 8×8 grid. I shows the final mapping of the blocks. (Courtesy of Doron Nussbaum, Paradigm Group, Carleton University)

integrate maps into the system. The maps should allow for the representation of heterogeneous fuel beds, relief features and weather conditions, which model the geographic conditions more accurately. Thus, the PFBP system allows a point fire in a cell to affect cells around it, thus spreading from one cell to the next. Since the input parameters can vary from cell to cell, using maps provides a more accurate model. The most significant difference is that the PFBP actually computes the propagation of fire from one cell to the next so that dependencies are consistent, i.e., a cell does not affect another if it cannot. It also accurately models the interaction between fires from different cells affecting the same cell.

Figure 4.2: Components of the Parallel Fire Behavior Prediction System

Integration of maps into the system requires additional work on the model since the interactions between neighboring cells must be defined. If a cell is burning, eventually the fire will cross the cell boundaries and ignite the neighboring cells, e.g., in Figure 4.3, we see the fire that has started in the center cell x has spread to the cells marked f . The cells marked u and f form the Moore neighborhood of cell x . Cell x is the center cell with the thick dashed border.

4.1.1 Inputs and Outputs

The FBP system accepts as inputs, the fuel type and weather conditions at a given point. The input also has a directional components such as the slope of the terrain and the wind. Currently, the wind speed is considered to be a constant across all

x - Igniting cell
f - affected cell
u - unaffected cell

Figure 4.3: Neighboring cells affected by fire spread

cells, although this may be trivially changed so that wind speed may vary from one cell to the next. When the FBP system is extended to process maps, it results in an anisotropic spread, i.e., one in which the rates of spread are not equal in all directions. If some directional components, such as the wind, are spatially homogeneous, we can represent them by a single variable. Components that are spatially non-homogeneous, such as the slope, require maps to be represented. The components that require maps to represent them because they are non-homogeneous are the ones that result in the anisotropic spread.

One of the outputs of the FBP system is the length of the major and minor axes of the ellipse as described in Section 3.4.3. To compute the exact shape and location of the fire due to directional components such as the slope and the wind, a translation and a rotation must be applied to the ellipse as illustrated in Figure 4.4. The dotted ellipse represents the original ellipse aligned to the horizontal axis; the dashed ellipse represents the ellipse after a translation, and the solid ellipse represents the ellipse after a rotation about the origin.

The FBP system models point fires with great accuracy and the fire accelerates for some interval of time until it reaches an equilibrium rate of spread. At the equilibrium rate of spread, the fire is not treated as a point source any more, but a line source.

The energy output of the fire per unit length along its front also becomes a constant.

Figure 4.4: Elliptical fire after translation and rotation

Thus, the parallel fire behavior prediction (PFBP) system developed here, extends the point based fire behavior prediction system (FBP) to work with rasters. Hence, it allows for the modeling of fires over heterogeneous fuel types and weather conditions. It takes into the account the result of long stretches of weather and their effect on the fuel bed. The inputs to the PFBP system are the maps representing the fuel type and weather conditions and the output is a composition of elliptic fires since the FBP system is based on an elliptic fire model.

As a result of the heterogeneity in the input maps, anisotropic patterns may be observed during the propagation of the fire from the source points. As it stands currently, the system supports only static weather patterns, but has the necessary hooks built in to support dynamic weather conditions. Dynamic weather conditions make the system much more complex since the cumulative effect of extended periods of weather must now be recalculated more often. Hence, it would require certain aspects of the FWI system 3.4.1 to be incorporated into the model. Since the FBP system is based on an elliptic fire model, the shapes of the fires observed represent a composition of ellipses.

4.2 Fire Propagation to the Neighborhood

The elliptical growth model used to model fire growth in the FBP system, as mentioned in Section 3.4.3, computes the growth of the fires along the major and minor axes of the ellipse. After a cell c_{ij} has been burning for a period of time and the fire has burned most of the cell, the fire may cross the cell boundary and affect the neighbors N_{ij} . This interaction between cells is referred to as *inter-cell interaction*. The accuracy of the inter-cell interaction often is the limiting factor in the accuracy of the entire model. Inaccuracies often cause either the predicted propagation path of the phenomenon or the final prediction result or both, to be wrong.

Due to the nature of spatial raster data, the paths that represent the propagation of various phenomena are artificially zigzagged [101]. Modeling the paths and propagation more accurately, involves the study of more than just the 3×3 (the Moore neighborhood) of a cell, e.g., the 5×5 neighborhood or a more complex irregular neighborhood. Since anisotropic media have irregular shaped, spread patterns, the neighborhood that is processed must be accordingly larger. Figure 4.5 demonstrates the irregularities generated by small neighborhoods, while modeling an ellipse shaped fire. These inaccuracies are a result of discretization of continuous terrain and discretization of continuous phenomena. As the neighborhoods being modeled became more sophisticated, these inaccuracies diminish, but the cost of computation increases dramatically.

Several approaches to handle inter-cell interaction are proposed in this section. The simpler approaches are faster in terms of computation but less accurate since they use approximations. The approximations may be done by either approximating the spread vectors to the eight reference directions, i.e., along the horizontal, vertical and diagonal directions, or by approximating the shape of the ellipse to its clipping rectangle, i.e., the size of the smallest rectangle that contains the ellipse. The intersection of the elliptic fire shape with the cell boundaries of the neighboring cells can also be used to compute inter-cell interaction. These approaches are outlined in greater detail in Sections 4.2.1- 4.2.3.

Figure 4.5: Errors due to simplistic neighborhoods

4.2.1 Vector Projections

The spread of the fire is calculated by the FBP system in four principal directions, i.e., the forward, backward and two flank directions, as shown in Figure 4.6. The solid line segment from the origin to the end of the ellipse along the major axis in the north-east direction represents the *head fire axis* while the line segment from the origin to the end of the ellipse along the major axis in the south-west direction represents the *back fire axis*. The solid line segment from the origin along the south-east direction represents the *flank fire axis*. The projections for the two flank directions are always symmetric.

In order to compute the effect of fire spread to the neighbors, the Moore neighborhood illustrated in Figure 4.3, is considered. One possible way to compute spread is to use the projections of the fire spread vectors onto the eight major reference axes as shown in Figure 4.6. Projections are made to the two nearest reference axes, one in the clockwise direction and one in the anti-clockwise direction, e.g., in Figure 4.6, the head fire spread is projected onto the north and the northeast rays of the axes. If the projections along the axes reach into the cell's boundary, then the cell is considered to be affected.

Figure 4.6: Projections of head and back fire onto the east-west axis

The angle of projection is easily computed as the absolute value of the difference between the reference and fire axes. Since the rate of spread in the head direction ROS_h and direction of the head fire spread are known, the angle α between the horizontal reference axis and the fire axis can be computed. Then the angle between the north-east ray and the fire axis is just $|\alpha - 45|$. The projection of the rate of spread of the head fire onto the east axis, ROS_{h1} and the north-east axis, ROS_{h2} is given by,

$$ROS_{h1} = ROS_h \cos \alpha \quad (4.1)$$

$$ROS_{h2} = ROS_h \cos |\alpha - 45| \quad (4.2)$$

These projections are indicated by the thick gray line in Figure 4.6. Similarly, the projection of the back fire vector onto the west axis, P_b is given by, $ROS_b \cos \alpha$ and is indicated by the thick black line in Figure 4.6.

$$ROS_{b1} = ROS_b \cos \alpha$$

Since the distance, D_i , to the centers of each of the neighbors is known due to the fixed size of the cells and the rate of spread along each of the axes is known (Equation 4.1), the time to ignition for the cell along the i^{th} axis, t_i is computed as follows:

$$t_i = D_i / ROS_i \quad (4.3)$$

Algorithm 1 : `compute_vector_spread` (*neighborhood, maps*)

Input: $m \times n$ neighborhood of the cell; input maps

Output: Computes the time when the neighbors will be affected

Time: $O(1)$

1. Read the map inputs for the cell.
 2. Call the FBP function and compute the point fire outputs, i.e., the rates of spread (ROS) along the head, flank and back axes.
 3. FOR each fire axis
 - (a) Compute the two nearest reference axes, one in the clockwise direction and one in the anti-clockwise direction.
 - (b) Project the fire axis onto both reference axes. ($R \cos \alpha$, where R is the rate of spread, α is the angle between the axis and the reference axis.)
 4. FOR each cell in the neighborhood that can burn
 - (a) Update the display and the rasters.
 - (b) Schedule each neighbor at the computed expected time of ignition.
-

Advantages

The advantages offered by the method of vector projection are:

1. **Speed**, since the FBP function is only invoked once to compute the rates of spread along the spread axes.
2. **Simplicity**, since the time to ignition can be computed by a simple linear relationship (Equation 4.3).

Disadvantages

Due to the simplicity of the projection method, there are two kinds of inaccuracies that may arise when using it. They are described briefly as follows:

1. Suppose the fire has spread from the center cell to the north and the northeast cell. If a projection is made from the northeast cell to the east axis, there will be an intruding projection into the east cell, indicating that the east cell has been affected, when in fact that is not the case.
2. The projection of the head fire onto the northeast axis in Figure 4.6 shows that the projection is a bit excessive and may show the fire to have propagated further than the true propagation distance. The true propagation distance is the distance from the origin to the intersection of the ellipse and the northeast axis. Similarly, sometimes the projection may be a bit too small compared to the true distance.

4.2.2 Bounding Box - Cell Boundary Intersection

An approximation often used for the area occupied by a simple geometric figure is its bounding box or clipping rectangle [42]. Hence, the elliptical shape of the fire may be approximated by the smallest rectangle it is completely contained in. In order to check if any of the neighbors are affected by the fire, one may check the intersection of the cell boundaries with the with edges of the bounding box.

If the ellipse is not aligned along the horizontal or vertical axes, but rotated as in Figure 4.7, then the bounding box is considered to be the rectangle obtained from taking the bounding box of the ellipse as if it were aligned along the horizontal or vertical axes and then rotating it through the same angle as the ellipse. One important difference between the bounding boxes used discussed in this context is that the bounding boxes are not always parallel to the north-south and east-west axes. The problem of computing the affected neighbors is reduced to the problem of testing the containment of any part of the rectangle's line segments in one of the neighbors.

Figure 4.7: Bounding box of a fire

Advantages

1. **Accuracy** : The results obtained by this method are very close to those obtained by using the method of ellipse intersection.

Disadvantages

1. **Complexity** : The calculations for this method, involve finding the ellipse, then its bounding segments and then the intersection with cell boundaries. Hence, it involves more calculations than finding the ellipse and cell boundary intersections.
2. **Speed** : This method is slower than the vector projection method and the ellipse intersection method while not offering more in the way of accuracy.

4.2.3 Ellipse - Cell Boundary Intersection

The most accurate test for the spread of the fire is the intersection of the ellipse representing the fire shape and the cell boundaries. Hence, after computing the elliptic

shape of the fire, the problem is reduced to the computation of the intersection of an ellipse and line segments. If the equation of the ellipse is known, and the cell boundaries are rectilinear then the time complexity of the finding a solution is that of finding the roots of a quadratic equation. All the cells whose boundaries intersect with the ellipse are affected are then scheduled for processing. The high level algorithm is described in Algorithm 3.

The general form of the equation of an ellipse is

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0. \quad (4.4)$$

An ellipse aligned along the X-axis, i.e., with its major axis along the X-axis has an equation of the form

$$Ax^2 + Cy^2 + F = 0.$$

After a translation through $(h, 0)$ is applied to the ellipse, the equation has the form

$$Ax^2 + Cy^2 + Dx + F = 0.$$

If the ellipse is rotated through an angle θ , then in the clockwise direction, then the equation becomes

$$Ax^2 + Bxy + Cy^2 + Dx + F = 0. \quad (4.5)$$

The equation of the rotated ellipse can then be used to find the intersection of the ellipse and the bounding line segments of the neighboring cells. Since the grid lines are rectilinear, the equation of the ellipse can be reduced to a simple quadratic equation in one variable. After the intersection points have been found, all the affected neighbors can be determined and scheduled for processing.

Advantages

- This is the most accurate of all the methods discussed in this thesis.
- It is simpler than the bounding box method and intuitive since it closely models reality.

Disadvantages

- It's only disadvantage is that it is more expensive in terms of computation than the method of vector projections. However, the gain in accuracy far outweighs the expensive computation.

4.3 PFBP Implementation Issues

The implementation of the PFBP system consists of three main modules described as follows:

1. PD_User : Interface to the propagation driver.
2. PFBP : The computation engine for fire spread to neighbors.
3. m FBP : The computation of point spread (within one cell).

The PD_User module must be implemented by every application that intends to use the propagation driver in order to manage event processing and scheduling. Hence the PFBP system implements this module in order to setup all the initial parameters for the propagation driver and to provide it with the application functionality it needs to determine how the propagation must proceed. The following functionality implemented by the PD_User module is needed by the driver to process and schedule events.

Time Stamp Comparison : The propagation driver maintains a queue of cells that need to be processed in order to propagate the phenomenon. This queue is a priority queue and the priority is determined by the time stamp. Hence, it needs to compare the time stamps in order to determine which cell is scheduled first. The cells which have time stamps closest to the current time will be processed first while those which have a time stamp at a more distant point in the future will be scheduled later.

Cell Comparison : Often, more than one neighbor may affect the cell. In such a case, it must be determined which neighbor should be allowed to affect the cell. This situation arises due to the data distribution scheme of the propagation driver. Any cell that is on the border of the block may be affected by a neighbor within the block or by a neighbor from a neighboring block. In a situation where both try to affect the same cell, it must be resolved which one affects the cell first.

Application Initialization : This part sets up the various parameters needed for the propagation driver to function correctly. It sets up the neighborhood size, the input and output maps, the seed events, background maps etc. It also sets up various parameters needed by the PFBP and FBP modules as well, such as the fire coefficients.

In the PFBP model, the time-stamps consist of two values, The first value indicates the time at which the cell becomes a potential hazard, i.e., one of its boundaries have been touched by fire while the second value indicates when the cell has ignited, i.e., the fire has reached the center of the cell and the cell is burning at a rate to potentially be a source of danger to its neighboring cells. Since the time stamp is not a simple numerical quantity it is the responsibility of the application programmer to provide the functionality which compares time stamps. Hence, the cells which have time stamps closest to the current time will be processed first while those which have a time stamp at a more distant point in the future will be processed later.

When processing a cell on the border of a block, sometimes it has to be determined if a cell will be affected by a neighbor in its block or from its neighboring block. In order to determine this the application programmer must provide a mechanism to determine which cell's effects are more significant. In the PFBP system, this is done by determining the ignition time of the neighbors. The ignition times are stored in one of the working rasters and the neighbor with the earliest ignition time is the one that affects the cell.

The setup segment of this module provides the propagation driver with information such as the size of the neighborhood. Thus, if the application needs the neighborhood of the cell to complete the processing of the cell, the neighborhood provided by the propagation driver is of the right size. Also, if the input maps must be accessed, the names of the input maps must be provided so that the data in the maps may be accessed easily. It also inserts the seed events (startup events) into the queue.

The PFBP module is the part that actually computes how the fire spreads from one cell to the next. The propagation driver hands off the seed events to the PFBP system and the processing of these events results in new events being scheduled. This process is reiterated until there are no more events or the maximum time has been exceeded. The maximum time is not an absolute measure of time, rather the simulated time, e.g., if the fire spread is only to be forecast for the next 6 hours (simulated time), the maximum time would be set to 6 hours.

The PFBP module can also be divided into three main functional units described briefly as follows:

1. Event Handling
2. Spread Calculation
3. Computing Ignition Times

The PFBP system is first activated with the source points of the fires and the times at which they will ignite. The event handling mechanism processes these events and then computes the effect of these cells on their neighbors. In other words, it determines for each neighbor of the current cell, when that neighbor will be affected by the current cell. Each neighbor will enter one of three states as follows:

1. Yellow : A cell enters this state when its border has been ignited by a neighbor.
2. Red : A cell enters this state when its center has caught fire and it becomes a potential hazard to its non-burning neighbors. A cell enters this state only after it has reached state Yellow.

3. Burnt : A cell reaches this state after the fire has ignited the boundary of the cell in the direction of the wind. A cell only enters this state after it has reached state Red.

In order to determine when a cell's neighbors are affected, the rate of spread of the fire through the cell must be computed. This done by calling the FBP system and determining the rate of spread within the cell. There are a couple of assumptions made to simplify the problem from the point of view of the PFBP system.

- The rate of spread of the fire is the *equilibrium rate of spread*.
- The fuel bed does not change from boundary to boundary, but from the center of the cell to the center of the next cell.

Both these assumptions are not unreasonable. The rate of spread of a fire accelerates for a certain amount of time and is constant thereafter if the fire being modeled starts from a point source. However, if the fire has been burning for a while and is modeled using a line source, the rate of spread may be considered to be a constant. Since the fires that were studied in this thesis were over extensive areas, they were correctly treated as line fires. The second assumption is a trivial assumption in that the point of recomputation is at the center of the cell. Hence, it is only at the center that the new fuel type is taken into account and it is considered to extend up to the fuel type of the neighboring cell. There is no recomputation that takes place when the fire reaches the boundary of a cell with a different fuel type.

The PFBP system calls the FBP system to determine the rate of spread within the cell. The FBP system returns a number of values, of which the most interesting are the length of the major and minor axis of the ellipse representing the shape of the fire. Using these two axes as the input, the equation of the ellipse, centered around the origin is computed. This equation is then translated so as to make the origin the source of the fire and then rotated to take into account the effect of the wind direction.

The equation of the rotated ellipse can then be used to find the intersection of the ellipse and the grid lines (bounding line segments) of the neighboring cells. Since the grid lines are vertical or horizontal, the equation of the ellipse can be reduced to a simple quadratic equation in one variable. Once all the intersection points have been found, all the affected neighbors can be determined and scheduled for processing.

4.4 Testing

The correctness of the algorithm underlying the PFBP system is proved quite easily. First of all, let us assume that all the inputs to the algorithm are in the correct range. Then the only processing that is done by the algorithm is to call the FBP system to predict the state of the in a cell and then to compute the intersection of this fire with the next cell. The FBP system described in Section 3.4.3, is a system whose correctness has been verified experimentally and theoretically and is outlined in [49].

The component of the PFBP that computes the fire spread does so as follows. The FBP system returns the length of the minor and major axes of the elliptic fire. The PFBP system then uses these to compute the equation of the ellipse as described in [65]. The ellipse is then translated and rotated to obtain the final equation using the techniques described in [65]. Then the intersection of the ellipse and the bounding lines of the neighboring cells are computed by substituting the value of x or y in the equation. This results in a quadratic equation which can then be solved to get the intercepts. This determines which of the segments has been affected. Since, each of these steps is correct, we state that the algorithm is correct.

The correctness of the scheduling etc. of the PFBP system follows from the correctness of the scheduling of the propagation driver. There are some issues that must be taken into careful consideration when using the propagation driver since the intermediate results presented by the propagation driver may have some causality errors as explained in greater detail in Chapter 6. However, the final results are correct.

Various tests were run to test the performance of the PFBP system. The platform that the PFBP was run on is the AVX Series II parallel computer made by Alex Informatique. The computer has 64 nodes and some additional processors to provide external access [81]. Each node consists of a compute engine (Intel i860 vector processor) and a communications processor (Inmos T805 transputer). This has been described in greater detail in Section 2.3.

The tests were run in order to determine the speedups achieved by allocating more processors to work on the problem, to test the speedups achieved for different image sizes etc. The results of these tests are discussed in Chapter 6. This application is also significant since it was the first application that used the propagation driver and was not written by the author of the propagation driver. Hence, it helped to uncover some problems with the driver, such as the displaying only some colors (the first 64 in the palette), providing a better interface to access maps and the ability to easily send parameters to the node programs.

Algorithm 2 : `compute_bounding_box_spread` (*neighborhood.maps*)

Input: $m \times n$ neighborhood of the cell; input maps

Output: Computes the time when the neighbors will be affected

Time: $O(1)$

1. IF the perimeter of the cell has been ignited
NOTE: The cell's perimeter has caught fire and it is not going to affect its neighbors yet (state YELLOW).
 - (a) Update current cell on the display and rasters.
 - (b) Reschedule the cell for a transition to state RED.
 2. IF the cell's center has been ignited
NOTE: The cell's center has caught fire and we will now consider it to be the origin of a new fire burning at its equilibrium intensity (state RED).
 - (a) Get the map inputs for the current cell.
 - (b) Update current cell on the display and rasters.
 - (c) Call the FBP function and compute the point fire outputs, i.e., the rates of spread (ROS) along the head, flank and back axes.
 - (d) Use these to compute the bounding box size.
 - (e) Compute the times that the neighbors will be affected.
 - i. Transition to YELLOW occurs when the ellipse intersects with the perimeter of the cell.
 - ii. Transition to RED occurs when the center of the cell is contained inside the ellipse.
 - (f) FOR all neighbors that are unburnt fuel.
 - i. Schedule the cell for a transition to state YELLOW.
 - ii. Update the rasters with the state and ignition times for the neighbors.
-

Algorithm 3 : `compute_elliptic_spread` (*neighborhood, maps*)

Input: $m \times n$ neighborhood of the cell; input maps

Output: Computes the time when the neighbors will be affected

Time: $O(mn)$

1. IF the perimeter of the cell has been ignited
NOTE: The cell's perimeter has caught fire and it is not going to affect its neighbors yet (state YELLOW).
 - (a) Update current cell on the display and rasters.
 - (b) Reschedule the cell for a transition to state RED.
 2. IF the cell's center has been ignited
NOTE: The cell's center has caught fire and we will now consider it to be the origin of a new fire burning at its equilibrium intensity (state RED).
 - (a) Get the map inputs for the current cell.
 - (b) Update current cell on the display and rasters.
 - (c) Call the FBP function and compute the point fire outputs, i.e., the elliptical growth outputs.
 - (d) Use these to compute the equation of the ellipse.
 - (e) Compute the times that the neighbors will be affected.
 - i. Transition to YELLOW occurs when the ellipse intersects with the perimeter of the cell.
 - ii. Transition to RED occurs when the center of the cell is contained inside the ellipse.
 - (f) FOR all neighbors that are unburnt fuel.
 - i. Schedule the cell for a transition to state YELLOW.
 - ii. Update the rasters with the state and ignition times for the neighbors.
-

Figure 4.8: Fire progression in the PFBP System. The exterior colors (thin red and yellow bands) correspond to the fire states while the interior bands (including the interior red and yellow bands) correspond to the time range in which the cells caught fire. In (b), one of the processors has not communicated its data to the display driver but it has done so by (c).

Chapter 5

Skeleton Finding

The skeleton was introduced by Blum in [7], as the representation of a closed boundary shape by idealized thin lines to retain the connectivity of the original shape. The skeleton of an object can also be defined as a thin line graph that preserves the topology of the object [18]. In general, one can define the skeleton of an image as a compact representation of that image such that it preserves the topological properties of the objects in the image and any part of the resulting skeleton has unit thickness. More specifically, the skeleton is the locus of all points p_s , such that p_s is equidistant from two or more distinct points p_{c_i} on the contour and the distance of p_s to these points is the minimum distance of p_s to any point on the contour.

The thinning of elongated objects is a fundamental preprocessing step in pattern recognition and image processing [51]. It has been often used as a compact representation for shape description, region interpretation and object recognition [18]. The greatest advantage of using skeletons is that they preserve the topology of a given image while reducing the total amount of data. This allows for the data to be processed more quickly, resulting in faster turnaround times to process complex objects.

Image skeletons have a variety of uses for image understanding in robotics, biomedicine and remote sensing [72], for modeling of objects [90]. They are also used

for other applications such as character recognition, chromosome analysis and generally for simplifying the processing of elongated objects since the width of the objects contains little or no useful information [100]. With the surge of use of GIS by various resource management organizations, image processing techniques such as skeletonization have direct applications in the realm of GIS.

One of the most interesting possibilities for usage of skeletons in the GIS domain and in the context of this thesis, is in fire management. In Figure 5.1, three elliptic fires and the skeleton of the area between them are depicted. The skeleton of this area represents the lines which will be the last to burn. Hence, if any rescue operations need to be performed, the paths along the skeleton represent the safest paths to areas interior to the fires. As well, if the skeletons are too close to the fire boundaries, rescue operations may not be performed in such critically affected areas endangering life and equipment of the fire fighting crews. Thus, used in concert, the PFBP and skeleton finding applications present promising solutions for fire management.

Figure 5.1: Quench points between elliptic fires

There are a number of approaches taken to finding the skeleton, of which two of the most common are listed below:

Thinning : Thinning algorithms find the skeleton of an image by repeatedly deleting pixels from the contours of an image until the image consists of paths of unit thickness.

Medial Axis Transform : The medial axis skeleton is defined as the locus of points where wavefronts propagating inward from the bounding contour meet.

More precisely, thinning is the process whereby cells that belong to the foreground, i.e., are part of the image, and are not critical to expressing the topographical features of the image are removed from the boundary pixels. This process is reiterated until there are no new pixels to be deleted, i.e.,

- All the pixels form a representation of the original image that is of unit width.
- The removal of a pixel will cause a discontinuity in the representation or cause unconnected components to be connected, thus modifying the topographical features.

The medial axis skeleton is found by propagating wavefronts from the bounding contour of the image and deleting all the points in the path of the wavefront such that:

- Each point on the skeleton is equidistant from two or more points on the bounding contour.
- There are no points on the contour closer to the skeleton point than these equidistant points.
- Each skeleton point lies in the interior of the bounding contour.

5.1 Exploiting Parallelism in Skeleton Finding

The earliest approaches to thinning were sequential and are interesting in that they generate consistent topologically accurate skeletons. However, parallel algorithms are

of particular interest since they allow for efficient processing using parallel computers [51]. A parallel thinning algorithm is one that uses a parallel *operator* and applies it to every pixel simultaneously during each iteration. In order to minimize the cost of communication, the neighborhood must be chosen to be as small as possible. For most fully parallel thinning operators a minimum support size of 3×3 is required [51]. In practice, however, the size of the neighborhood does not make a significant difference since the cost of communication is mostly the startup cost. The cost of sending a 5×5 instead of a 3×3 neighborhood is negligible.

Skeleton finding in raster images is a problem that lends itself very well to being parallelized. The nature of the input image allows it to be segmented easily and for various parts to be processed independently. However, during processing, great care must be taken to prevent discontinuities in the input image. Methods to address this are discussed in Section 5.2.1- 5.2.2.

5.2 Skeletons of Binary Images

A binary image is an image that is composed of binary pixels, i.e., the pixels can take the value 0 or 1. All pixels that have the value 0 are considered to belong to the background while pixels with value 1 are considered to be the foreground. An *object* in the image is a set of connected foreground pixels. In order to find the skeleton of the image, all the foreground pixels that are not vital to the representation of the figure's topological information are changed to belong to the background.

While finding the skeleton of an image, some operators only allow foreground pixels to be converted to background pixels. Such operators are known as reduction only operators. Most parallel thinning operators are reduction only operators. All operators must confirm to the following connectivity criteria during all iterations [53]:

- An object in the source image S (a connected set of ones) should not be disconnected into two or more distinct objects.

- An object in S should not be deleted.
- Distinct objects in S should not be merged.

The high level algorithm for this is very simple. During each iteration, each pixel is tested to check if its deletion violates the connectivity criteria. If it does, then it is saved otherwise it is deleted. If there were no pixels deleted during the last iteration, the algorithm terminates. If any pixels were deleted, then the algorithm is re-applied to the pixels that were not deleted in the previous iteration. The crux of the problem is in determining the criteria for deletion of pixels. These criteria should not violate the conditions mentioned above.

5.2.1 Thinning Method

As discussed earlier, a thinning algorithm reduces an image so that each *stroke* (extension) of the image is of unit thickness. This reduction should not deviate significantly from the centers of these features as shown in Figure 5.2. Figure 5.2(a), shows the skeleton located correctly towards the middle of the object, while Figure 5.2(b) shows a skeleton that is incorrectly located towards the boundary as opposed to the middle. Most algorithms work with a function of the 3×3 neighborhood of a cell to compute the value of the cell for the next iteration [31]. The discussion that follows sheds light on the algorithm presented in [31], which is based on work presented in [91].

Cells are removed if they are on the edge of a stroke. Cells are considered to be on the edge of the stroke if they have between two to six connected neighbors, where connectedness is defined as the existence of a path between between two true neighbors than only passes through true neighbors. Hence, to determine if the cell is on an edge, pairs of neighbors around the cells are studied. The existence of a (0,1) pair means that traveling in a clockwise direction around the neighbors, a cell encountered has value 0 and the cell encountered immediately after it has value 1. It is observed that there are between two and six neighbors if the following conditions are met:

Figure 5.2: Correct (a) and incorrect (b) locations for a skeleton

- At least one (0,0) pair is found
- At least one (1,1) pair is found
- Exactly one (0,1) pair is found

This is evaluated for every rectilinear triple of neighbors so that the neighbors are checked in the following order: (NW, N, NE), (NE, E, SE), (SE, S, SW), (SW, E, NW). All the (0,0), (0,1), (1,1) pairs are aggregated for all the neighbors, i.e., the four triples and then every cell that belongs to the foreground is on an edge if it satisfies the above mentioned properties. Thus, the cell is a candidate for survival to the next iteration if

$$vC \wedge \sim edgeC \quad (5.1)$$

where vC is the value of the cell and $\sim$ is logical NOT. In order to make sure that the parallel algorithm is correct (NOTE: this is not required for the sequential version), it should satisfy the following constraints.

- The removal of the edges of the stroke should not cause the entire stroke to be removed.
- All four elements of a 2×2 stroke should be kept.

- The connectedness of the overall structure should not be affected by the simultaneous removal of a number of elements.

A common solution is to introduce a directional bias and/or sub iterations. A directional bias may be introduced by favoring a direction over other and minimizing the resultant distortion by introducing sub-iterations that differ only in the direction they favor. Applying this principle to Equation 5.1, and favoring the south or east edge or corner elements for removal yields the following expression:

$$\sim vE \vee \sim vS \vee (\sim vN \wedge \sim vW) \quad (5.2)$$

Combining, the negation of Equation 5.2 (to get the equation for survival) and Equation 5.1, yields:

$$vC \wedge (\sim \text{edge}C \vee (vE \wedge vS \wedge (vN \vee vW))) \quad (5.3)$$

The equation for the second sub-iteration is obtained by interchanging north with south and east with west. Hence, we get the expression

$$vC \wedge (\sim \text{edge}C \vee (vW \wedge vN \wedge (vS \vee vE))) \quad (5.4)$$

Processing With Larger Neighborhoods

The survival of an element into the next iteration may be computed in only one iteration if a neighborhood larger than 3×3 is used. In the above algorithm, for instance, the survival of a cell can be determined within one sub-iteration if a 5×5 support is used.

Considering realistic scenarios of inter processor communication the overhead of using a 5×5 neighborhood as opposed to a 3×3 neighborhood is negligible. In fact, the linear model of communication which states that the cost of communication has a constant startup cost and a linear cost associated with the amount of data being transferred, indicates that neighborhood communication should be batched effectively in order to avoid the startup overheads associated with sending one neighborhood at a time.

A two cell wide vertical stroke is kept by keeping one of its edges. Hence, if the westward bias is maintained, the west edge is preserved if it is not a corner and the east neighbor is on edge, i.e.,

$$edgeE \wedge vN \wedge vS$$

The north edge of a horizontal two cell wide stroke is preserved analogously if

$$edgeS \wedge vE \wedge vW$$

The removal of 2×2 square is prevented by checking if the east, south east and south neighbors are on corners as indicated by

$$edgeE \wedge edgeSE \wedge edgeS$$

Incorporating all the above equations with Equations 5.3- 5.4 yields

$$\begin{aligned} vC \wedge (\sim edgeC \vee (edgeE \wedge vN \wedge vS) \vee \\ (edgeS \wedge vW \wedge vE) \vee \\ (edgeE \wedge edgeSE \wedge edgeS)) \end{aligned} \quad (5.5)$$

Due to the compactness and simplicity of this parallel operator, it was chosen to be implemented for the skeleton finding.

5.2.2 Fire Fronts Method

The original method to find skeletons, proposed in [7] uses grass-fire propagation. The objects in the image are considered to be composed of grass. If the perimeters of the objects are lit, the fire would burn and propagate towards the interior of the object. The propagating fire fronts would meet and extinguish each other. The locus of these quench points then forms the skeleton of the object. The quench distance is the distance of a quench point from the contour, as illustrated in Figure 5.3.

The method described in this section is proposed by [100]. It uses the property of wave propagation for skeleton finding. This technique is based on the contour

following principle and simulates the fire propagation technique. It is more efficient in that it works only on the shape's boundary, which avoids going back to check the area already burned. The exterior contours of the image are traversed in the counter-clockwise direction while the interior contours are traversed in the clockwise direction.

Figure 5.3: Propagating fire fronts and quench lines

5.3 Skeletons of Gray Scale Images

Skeleton finding has been studied mainly for binary images on the assumption that they can adequately represent gray scale images, but increasingly more interest is being expressed in finding the skeletons of gray-scale images since the distortion due to conversion from gray-scale to binary images is avoided [3]. As well, often the different gray levels represent meaningful information, as in biomedical specimens.

There are several approaches to finding the skeleton of a gray scale image. The Gray Weighted Distance Transform (GWDT) obtained by Levi and Montanari [66] is derived by associating to each point the length of the minimal path from that point to the back ground (in effect a non increasing path if the gray level is considered to be the elevation of a pixel). Another approach based on *shrinking*, *expanding* operations called the Min-max Medial Axis Transform (MMAT) is proposed in [77]. The MMAT

algorithm does not require segmentation of the image into objects and foreground, but it does not guarantee connectivity or thinness.

Dyer and Rosenfeld, in [30], developed a generalization of a thinning algorithm for binary images and applied it to gray scale images. By their criteria, two points P and Q are connected if and only if the gray value of any element along the path that connects them is not less than both of them. However, this method has the drawback that the local connectivity does not guarantee global connectivity. Thus, it would not lend itself well to being parallelized. As well, the skeleton is placed in a central position with reference to the image boundary, and not along the regions of high intensity.

5.3.1 Iterated Pixel Removal method

In [3], Arcelli and Ramella describe an efficient technique to accurately find the skeleton of gray scale images. As mentioned earlier, the gray scale image can also be considered as an elevation map. The gray skeleton of this map can then be related to topographical features such as ridges, peaks etc. [3].

The exact definitions of the terms can be found in [3]. A brief explanation of the terms is provided to facilitate understanding of Algorithm 4. A *hollow* is a region which has a border of higher elevation pixels around it, while a *plateau* is a region that has a border of lower elevation pixels around it. Plateaus and hollows have associated heights and depths defined as the path from the region to the back ground (or a pixel of value k) such that the path is non-increasing for plateaus and non-decreasing for hollows.

A major problem encountered when dealing with gray scale images is the finding of ridge lines when there are many hollows close together. Hollows are surrounded by ridge lines (by definition), and removal operations cannot remove ridge lines, the pixels inside a hollow can never be accessed. Thus, some regions of the image may not be correctly thinned. Arcelli and Ramella, are the first ones to propose a viable solution

to this problem. An application dependent parameter known as the *significant depth* can help to avoid this problem. All hollows that have a depth not smaller than this depth are assigned to the background therefore reducing irrelevant information.

The iterated pixel removal method permits the handling of gray-scale patterns including hollows provided that their significant depth in the problem domain is known, and produces a one pixel wide skeleton that tends to be placed in the higher intensity regions or the mid-lines of the constant intensity regions. As mentioned in the introduction to Section 5.3, these are both extremely desirable characteristics in gray skeletons.

Algorithm 4 : find_gray_skeleton (*image*)

Input: Image of size $m \times n$

Output: Skeleton of the input image

Time: $O(mn)$

1. Smooth F using *min*, *max* functions
 2. Fill in or flatten insignificant hollows and plateaus
 3. Apply the removal operator and rank the gray values the removed pixels
 4. If no border points are used, go to step 8.
 5. Remove hollows whose pixels are less than the maximal ranked gray value
 6. Apply the restricted removal operation, to successively remove sets of pixels with gray value not greater than each of the ranked gray values. Continue this till all gray values up to the maximally ranked gray value are removed.
 7. If at least one border point is removed, go to step 3.
 8. If there are hollows, remove all of them and go to step 3.
 9. The un-removed pixels are skeleton pixels.
-

5.4 Parallel Skeleton Finding

The skeleton finding application addressed in this thesis is based on the method described in Section 5.2.1. There were several factors taken into account before deciding the path taken for the parallelization of this problem. The skeleton finding problem needs a 3×3 or a 5×5 neighborhood in order to determine whether or not a pixel survives for the next round of deletions. This suggests that the cellular automata driver component of NEMO would be a likely candidate since it can provide neighborhoods of a cell and it provides synchronization at the end of every iteration.

A counter argument to the above implementation choice is that only the cells that are on the contour of the image are candidates for deletion. The cells that are on the interior of the image cannot be deleted. Hence, it would be much more efficient instead of processing each cell, to only process the current active border at any given time. Since the propagation driver provides the concept of an active border, it also seems like a likely tool for parallelization. However, the propagation driver does not provide distinct iterations and thus no periodic synchronization.

Figure 5.4: Incorrect deletion of active border pixels in the propagation driver

The cellular automata driver uses a simple partitioning scheme to distribute the data, which works very well for an application such as skeleton finding since the image

can be evenly partitioned and then each segment of the image can be relatively independently processed. The processed segments can then be combined to reconstruct the original image. The higher overhead of the more complex partitioning scheme implemented by the propagation driver is not required for this application.

In Figure 5.4, a possible problem that may arise when using the propagation driver is highlighted. The incorrectly deleted pixels are deleted after what would be iteration 1 on the cellular automata driver. In the propagation driver since there are no iterations, suppose that all the pixels on the outermost border are scheduled correctly. After this iteration, there is a new active border. Then, the pixels that are eligible for deletion are deleted right away without waiting to synchronize with other processors. This can cause the incorrectly deleted pixels to be deleted since they satisfy the criteria for deletion. Thus, since synchronization after each iteration is necessary for the algorithm to function correctly, the cellular automata driver was chosen as the tool for parallelization. The implementation of skeleton involved three distinct phases:

- Sequential implementations
- Parallel implementations
- Optimizations

The sequential implementations are based upon the details discussed in Section 5.2.1 while the optimizations are discussed further in Section 6.2.2.

5.4.1 Parallel Implementation

As mentioned earlier in this section, the cellular automata driver was chosen to be the parallel tool for the implementation of skeleton finding. The architecture of the skeleton finding system is shown in Figure 5.5. The cellular automata driver provides parallel services such as:

- Initial allocation of data to the processors
- Neighbor data access from other processors

- Synchronization, communication to display driver

However, it is the responsibility of the application to determine how each cell must be processed, what results must be displayed after every iteration. The parallel component of the skeleton finding system is minimal and acts as the glue between the cell driver and the edge detection/pixel deletion component.

Figure 5.5: Skeleton Finding in the NEMO framework

The parallel component consists of the following components, implemented as per the requirements enforced by the cellular automata driver:

Initial Setup : During the initial setup, first the neighborhood information such as the size of the neighborhood etc. is setup and the raster initialization is invoked. Then, the first round of exchange of neighborhood data takes place. This allows each processor to perform any preprocessing if required with the correct neighborhood.

Raster Initialization : Memory is first allocated for spatial data on the current processor and all the spatial data that is to be received at the start of each

iteration from the neighbors. Then all the raster block information is setup, and a call to the raster manager populates the buffers allocated for spatial data.

Transition function : The transition function is the function is called once every iteration and determines the state of the cells for the next iteration. This is the part that calls the edge detection and deletion component.

Next Iteration Setup : During this phase, each processor sends the neighborhood data to each of its neighbors and decides whether or not the data should be output to be displayed during this iteration.

Conversion to Display Data : At the end of every iteration, if the data is to be displayed, then it should be converted to a format that the display manager can decipher. This function performs any filtering or post-processing necessary for the image to be displayed.

Earlier versions of the cellular automata driver did not provide a mechanism for the access of neighbor data via an automatic function. It was required that the code to access neighbor data from other processors be provided by the application programmer. Hence, a generic scheme for providing a neighborhood of any size was provided by the skeleton finding application.

In Figure 5.6, the neighborhood of the dark cell *c*, on the north-east corner, comes from spatial data of the processor on which *c* resides, indicated in gray. The neighborhood is also composed of data from the north, north-east and east neighbors of that processor. This is represented by the square with the thick border around the black cell *c*. Thus, in order to find the neighborhood of a cell it may be necessary to look at the data *block* from one or more of the neighboring processors.

The data in each of the blocks is arranged in a single array whose size varies as the number of neighboring cells in that block, e.g., the north block has *raster_width* × *neighbourhood_height* elements, while the north-east block has *neighbourhood_width* × *neighbourhood_height* elements. Thus, building a neighborhood consists of two phases.

Figure 5.6: Location of neighbors during skeleton finding

The first phase determines the intermediate index of each neighbor as though it were on the current processor. These intermediate indices could be out of bounds. The intermediate indices are then used to compute the *block*, e.g., the NORTH block, and the index of the value within the block.

Once the neighborhood is built, the deletion function is called with the neighborhood as the argument. The deletion function calls another function to determine if the pixel is on an edge and if it is, it applies certain criteria to determine if the pixel can be deleted. If removing the pixel changes the connectivity or characteristics of the image, it is saved into the next iteration. The details of edge detection and pixel deletion are described in Section 5.2.1.

5.5 Testing

The correctness of the skeleton finding algorithm in Section 5.2.1 is discussed in detail in [53]. The implementation of skeleton finding is one whose correctness is easily verified for the test data, by visual inspection. On startup, the cellular automata

driver takes as input, an update period, specified in number of iterations. If this is set to one, the raster is updated every iteration. Hence, visual inspection for the absence of disconnection, deletion of non edge pixels (introduction of holes) and the non-deletion of edge pixels on thick branches, can show that the skeleton for the test images is correct.

Another possible method to verify the correctness is to reconstruct the original image from the skeleton and the deletion data. Simple reconstruction of the image by starting from the skeleton and then reinserting the deleted pixels will obviously reconstruct the image correctly even if a pixel was deleted too early or too soon, i.e., in the wrong iteration. A more correct method would be to start re-inserting the pixels in the reverse order of deletion. Consequently, pixels that were deleted last are the first ones to be reinserted. Then, the algorithm recreates the neighborhoods of the reinstated pixels to verify that all the newly added pixels would indeed be deleted. Currently, the parallel version does not verify itself but a sequential implementation has been done and should require minimal effort to be parallelized.

Image reconstruction has the potential to be very useful when trying to recreate original images after complex processing operations. An example of possible use of reconstruction in the domain of GIS is as follows. If two maps are being overlaid, one possible way to do this is first to find the skeletons of all the objects on this image. It is easiest to overlay the skeletal images and then reconstruct the images in order to get an overlay of the original images. During the reconstruction, the maps can be realigned after every iteration of reconstruction in order to have the best possible overlay.

In order to study the parallel performance of the skeleton finding system, the test platform was the same as the one described for the PFBP system in Section 4.4. Section 2.3 describes the parallel platform in greater detail. The tests were run to determine the speedups achieved by allocating more processors to work on the problem, to test the speedups achieved with different image sizes etc. The results of these tests are discussed in greater detail in Chapter 6. It uses a larger neighborhood than the default von Neumann or Moore neighborhoods since it requires a 5×5

neighborhood. Hence, data must be exchanged between diagonal neighbors as well. This application helped to identify some problems and shortcomings of the cellular automata driver which have been addressed since.

Figure 5.7: Progression of skeleton finding (thinning) in an image with roads, rivers and lakes. Note: The number of connected components is the same in the original image (a) and the final image (d) after 20 iterations.

Chapter 6

Results and Performance Analysis

The purpose of this thesis was to describe the development of GIS applications within the scope of NEMO. Chapters 4 and 5 describe in detail the reasons for using different tools within NEMO, such as the propagation driver and the cellular automata driver. One of the principal reasons for the development of NEMO is to achieve improvements in performance by harnessing the potential of parallel computing.

The key measurement that provides insight into performance improvement is the *speedup*. The speedup $S_p(n)$ is defined as the improvement in performance by using p processors instead of a single processor, for a problem with an input size of n . Thus, if $T^*(n)$ is the sequential complexity to solve the problem and $T_p(n)$ is the time taken by p processors, then the speedup is defined in [60] as

$$S_p(n) = \frac{T^*(n)}{T_p(n)} \quad (6.1)$$

In order to compute the true gain, the best known sequential time on the best sequential hardware may be used instead of $T^*(n)$, but this does not provide an insight into the gain by parallelizing the problem. $T^*(n)$ must be obtained by running the sequential algorithm on hardware that is comparable in performance to a single node

in the parallel hardware. This is especially true since state of the art parallel machines do not always use the most recent sequential processors.

There is usually a practical limit on the speedup that may be achieved by adding more processors. This limit is due to the amount of time spent in communication or waiting for data from other processors as described in detail in Sections 2.2.1- 2.2.3. An ideal parallel system boasts a speedup equal to or greater than the number of processors, i.e., where $S_p(n) \approx p$. Super-linear speedup is also possible due to the generally larger memory present in parallel computers, fewer cache misses due to better data distribution etc.

Another indicator of performance is the *efficiency* and is defined by [60] as

$$E_p = \frac{T_1}{pT_p} \quad (6.2)$$

It is a measure of the effective utilization of the p processors for the given algorithm. If each processor is doing useful work, all the time, then pT_p should be the time taken to complete the job. The time taken by a single processor is T_1 and should be roughly equal to pT_p . Thus, an E_p value of approximately one indicates that the algorithm will run p times faster for p processors.

There is a drop off in a performance after a certain point, no matter how many more processors are added to solve the problem. This fastest running time achievable through parallelism is denoted by T_∞ . Thus, $T_p > T_\infty$ for all values of p . Hence, if we keep increasing p beyond T_∞ , we get a drop-off in the efficiency, but no improvement in the running time. Practically speaking, beyond this point the performance actually starts to get worse due to communication and load balancing overheads.

6.1 Performance gains in the PFBP

Any performance gains that can be achieved for the PFBP system depend heavily on the communication and the load-balancing achieved by the propagation driver.

In order to fully understand the performance gains, it is necessary to have some knowledge of the propagation driver implementation.

6.1.1 Propagation Driver

The propagation driver maintains a priority queue of cells to be processed. The cells are prioritized by the time at which they are scheduled for processing. The cell to be processed earliest has the highest priority. Each processor maintains its own priority queue. Hence a cell which is scheduled for processing at simulation time $t = 500$ on processor μ_a may be processed before a cell scheduled for processing at time $t = 450$ on processor μ_b . This may cause some causality errors to appear until μ_a and μ_b communicate and exchange data directly or through another processor [32]. After communication, the values for some cells may have to be recalculated in order to reflect a consistent global state.

Figure 6.1: PFBP Computation Time [Block size: 32×32 , 6 fires]

The propagation tries to achieve uniform load-balancing by performing data distribution so that each processor has roughly the same number of active cells at any time ideally. In order to achieve this end, it uses rasters which have information organized in blocks. The blocks can be allocated to processors based on a logical tiling scheme so that each processor gets a block and then this cycle is repeated until there are no more blocks. This achieves an accordion like folding over the processors as detailed in Chapter 4.

6.1.2 Analysis

Graph 6.1 shows that as the number of processors working on the problem is increased, the amount of time taken to arrive at the final result decreases very quickly at the beginning and then levels off. There are several factors that contribute to this. Not every processor is doing useful work at every instant. This means that either processors are idle waiting for the data, or spending a large amount of time communicating. Since most of the tests involved six fires, at best six processors were working at the beginning. Additional processors only start to work as the fire propagates out of the block into the block of the neighboring processors. As, a result, if a larger area is being studied, i.e., a larger raster with more fires, the expected speed up would be improved.

As seen in Graph 6.2, there is a gain in performance up to nine processors and then it drops off substantially. There are several reasons that can explain this rapid drop off in performance:

Locality: The spread computation function (modeling function) is very expensive in terms of computation. Hence, the fires take a long time before they propagate to the next processor. Since the fires are local to each processor it takes a while before the maximum amount of parallelism is exploited.

Location: Even if the fires are in distinct blocks on the map, both blocks may reside on the same processor, causing an uneven distribution.

Figure 6.2: Speedups achieved for the PFBP System

The above reasons suggest that for smaller block sizes the expected speedup is higher. Graph 6.3 exhibits exactly this property. The graphs the 768×768 and 640×480 rasters demonstrate radically different computation times for different block sizes. In addition to the earlier explanations of locality and location, one important aspect to be noted is the causality errors. Larger block sizes mean that it takes longer for a fire from a neighboring block to propagate to the current block. Hence, the time elapsed is longer and therefore the computation may be performed for each cell more than once. This observation is confirmed by the fact that each cell seems to get processed more often when using larger block sizes. the 640×480 graph, takes more time to compute than

In Graph 6.4, increasing the number of fires increases the amount of time required for computation. It does not seem to take greater advantage of parallelism. This is misleading since more cells need to be processed per processor in order to process a larger number of fires. A more informative relationship is that between the speedup and number of fires. The larger the number of fires the greater the speedup achieved

Figure 6.3: Effect of block size on running time [16 procs., 6 fires]

with a larger number of processors (as long as $T_p < T_\infty$).

When there are 8 fires, it takes the 640×480 raster longer to burn, than the same number of fires for the 768×768 raster. Consequently, in Graph 6.4, the curve for the 640×480 raster crosses that of the 768×768 raster. This seems counter-intuitive but can be explained by the load balancing resulting from the data distribution. In the first case (640×480), analysis of the input shows that two fires start on the same processor $\mu_{3,3}$ thus overloading that processor and making it the bottleneck.

6.2 Performance gains in skeleton finding

This section examines the effect of parameters such as the image size, number of processors etc. on the processing time for skeleton finding. In addition, experimental results indicate the need to characterize images in order to predict the nature of the achievable speedups for those images. In order to completely understand the

Figure 6.4: Effect of number of fires on computation time [16 procs., Blocksize: 32×32]

results of the tests, it is necessary to gain some insight into the functionality of the cellular automata driver. All tests were run for ten iterations since the cellular automata driver does not have a mechanism by which the application can notify it of termination.

6.2.1 Cellular Automata Driver

The cellular automata driver has a simple data distribution scheme. The default scheme for data distribution is to partition the raster into the same number of blocks as the number of processors. The size of each partition is approximately equal on every raster so as to distribute the load evenly. The data distribution scheme is outlined in greater detail in [81].

If the data distribution is to be any different than the default scheme, all data distribution, neighbor data exchange between iterations etc. must be managed by the application programmer. A possible extension of the cellular automata driver

would be the ability to provide different data distribution that are common in most applications, e.g., cyclic, block or hybrid schemes.

6.2.2 Analysis

The current implementation of the cellular automata driver and the skeleton finding application, requires that every cell that is part of the foreground, be checked for deletion. Checking that a candidate can be deleted is the only expensive part of the computation since actually deleting a pixel only requires one write operation. The algorithm explained in detail in Section 5.2.1, shows that only the pixels on the edge of the image that belong to the foreground are candidates for deletion. This indicates, that not all pixels need to be checked at every iteration, but only the ones on the edge.

There is a simple optimization that can greatly improve performance when finding the skeleton. Each iteration generates a new outer surface for the image as shown in Figure 5.4 (a). The new outer edge is formed from the area in contact, i.e., in the immediate neighborhood of the outer edge for the previous iteration. Thus, if during the first iteration, an active border is marked, the only cells that form the active border for the next iteration are those that become exposed due to deletion of the cells. These cells are then obviously, only those in the immediate neighborhood of the cells deleted in the current iteration.

Since the current implementation does not have this optimization, the number of cells in the foreground determines the computation time. Hence, we propose the term *workload* to denote the number of cells that determine the computation time. The workload ω_i of image is defined as the number of cells not deleted that belong to the foreground. These are all the internal cells and the cells on branches of unit thickness.

$$\omega_i = P_{nd}(i) \quad (6.3)$$

In Graph 6.5, the effect of the *maximal workload*, as in Equation 6.4 has a greater

influence on the computation time than the clustering in the image. The maximal workload is defined as follows:

$$\omega_{max} = \{\omega_k | \omega_k \geq \omega_i, \forall i\} \quad (6.4)$$

In fact, the standard deviation of the workload σ_ω , is not an indicator of the load-balancing achieved by the data distribution of the cellular automata driver. Even if the average workload is the same for two images, the determining factor in limiting the speedup is not σ_ω , but ω_{max} .

Figure 6.5: Computation time for uniform and clustered pixel distributions

Graph 6.6 demonstrates that the computation time goes up as the image size increases. This is expected, since the workload increases with the increased image size.

It should also be noted from Graph 6.7 that the slope of the curve for the larger image (1024×1024) is the steepest indicating a greater speedup. The slope for the 800×800 image is greater than that of the 640×640 image. This indicates that

Figure 6.6: Computation time for various image sizes

the larger the rasters being processed, the greater the speedup realized by harnessing parallelism.

Similar images with approximately the same number of cells on the boundary, but a different number of internal cells will take significantly different amounts of time to process. It would be useful to describe a property that expresses the relationship between the total number of foreground pixels and the number of pixels on the edge. In this thesis, we propose a metric to capture this quality in an image. We refer to it as the thickness τ_i ,

$$\tau_i = \frac{P_e(i)}{\omega_i} \quad (6.5)$$

where, $P_e(i)$ is the number of pixels on the edge of the image and ω_i is the workload for the image as defined in Equation 6.3.

Thus, if the above optimization is implemented, the expected speedup should be approximately equal to the thickness τ_i . We expect that images with a similar workload but different thicknesses should show different values of time. However,

Figure 6.7: Effect of the number of processors on computation time

with the current implementation, the workload is the determining factor since every cell is processed and the thickness has no role whatsoever. Hence, two images with the same workload but different thicknesses take the same amount of time.

An interesting quantity to measure would be the number of cells processed by the propagation driver and the cellular automata driver in unit time. Even though the propagation driver has a greater overhead in terms of data distribution and scheduling, it would process the fires much quicker than the cellular automata driver since it needs to process fewer cells to arrive at the result. Another interesting quantity to contrast would be the standard deviation of the workload σ_w and maximal workload σ_{max} .

Chapter 7

Conclusion and Future Work

The intent of this work was to demonstrate the potential offered by parallel computing to improve performance of two GIS applications. The applications, fire modeling and skeleton finding were implemented in NEMO to demonstrate this potential. These were the first applications developed to test the propagation driver and the neighborhood capabilities of the cellular automata driver, written by an independent user. A cellular automata application is being written, in parallel, by Zhengping Wang.

These applications helped to highlight various performance and stability issues and achieve resolutions to these issues in both drivers. They verified NEMO's client server architecture and initiated an effort to increase its robustness, as a result of which NEMO is a reliable platform to build GIS applications. Some of the features that were added or modified are as follows:

- The propagation driver provides the ability to change a cell's color without associating an event with that cell.
- The map accessing interface of the propagation driver is more compact and intuitive.
- The cellular automata driver now provides the ability to access the data of any neighboring cell using a single function.

- The cellular automata driver functionality to exchange data with diagonally neighboring processors has been made more robust.

The implementation tested the capabilities of NEMO, achieving significant speedups for both applications. It also identified potential bottlenecks in the performance and the characteristics of problems that might be identified ahead of time to enhance the performance of the parallel implementations of an application. These improvements are significant in that they highlight issues that the application programmer (not the parallel tools) can address to improve performance.

7.1 PFBP System

The PFBP system is the first realistic parallel fire modeling system that is not based on highly simplified fire models, but experimentally verified models such as the FBP system. It is of great interest to the fire modeling group at Great Lakes Forestry Center (GLFC), who have constantly provided input on the correctness to the parallel extensions in this model [98]. As well, a new physical model has been proposed by them in [99]. This model places greater demands on computational resources than the FBP system and is an ideal candidate for parallelization using the propagation driver. There is much interest in collaboration on this front, on the part of GLFC.

As it stands currently, the PFBP system supports only static weather patterns, but has the necessary hooks built in to support dynamic weather conditions. Dynamic weather conditions make the system much more complex since the cumulative effect of extended periods of weather must now be recalculated more often.

The PFBP system is easily extendible because of its highly modular design. All the functionality, that builds the structure to be sent to the FBP system, is isolated in one function. If the weather is to be obtained dynamically, the code to read the map simply needs to be replaced with the code to access dynamic weather data. The structure that is passed to the FBP system already has storage for all this data.

Currently, the PFBP system only shows the regions that are affected by fires and the time of burning of those regions. Since the FBP system already generates outputs for aspects such as the rate of spread, the heat output of the fire etc. it would be interesting to generate maps for these aspects from the PFBP system as well. This would be easy to do since the parallel extensions for fire spread could also be used for the maps for these aspects.

7.2 Skeleton Finding System

The skeleton finding problem appears to be propagation based. However, closer examination reveals that it is indeed a cellular automata type problem, as detailed in Section 5.4. The optimizations suggested in Section 6.2.2, implement the efficient solution of a propagation model on the cellular automata driver, albeit with a significant space overhead.

The skeleton finding system highlighted some shortcomings of the cellular automata driver within NEMO and to the consequent resolution of these shortcomings. It was also the first application to test boundary conditions in the cellular automata driver. The generic neighborhood provision scheme of the driver is based on an implementation first supplied by the skeleton finding system.

The current implementation of the cellular automata driver does not have a mechanism whereby the application can notify the driver that it has terminated. Consequently, the skeleton finding application is run for a number of iterations determined at startup. It would be very useful for an application, not the just skeleton finding, to instruct the driver that it has terminated and then have the driver terminate the program.

There are many possible ways to improve the skeleton finding system. It is of great interest to find the skeletons of the gray scale images. This has applications particularly in GIS, in object identification and matching and map overlaying. Some

of the primitives that are required for gray scale thinning have been implemented and this should be extended to a full implementation. As well, we are the first to propose the use of skeleton finding and the PFBP system in concert to generate fire management data, such as safest paths.

We have presented a pragmatic approach to some applied problems when trying to use parallel techniques in the GIS domain and their solutions. We also address the issues that arise when tackling such problems and suggest improvements from the tools and the applications perspective.

Bibliography

- [1] D.H. Anderson, E.A. Catchpole, N.J. de Mestre, and T. Parkes. Modeling the spread of grass fires. *Journal of the Australian Math Soc.*, 23(Series B):451–466, 1982.
- [2] D. A. Andonov, P. J. Bryanston-Cross, C. E. Towers, and S. T. Kitson. Gradual parallelisation approach for a fluid dynamics solver. In T. S. Durrani, W. A. Sandham, J. J. Soraghan, and J. Hulskamp, editors, *Applications of Transputer 3*, pages 365–370, Amsterdam, 1991. IOS Press.
- [3] Carlo Arcelli and Giuliana Ramella. Finding grey-skeletons by iterated pixel removal. *Image and Vision Computing*, 13(2):159–167, Apr. 1995.
- [4] C. Aykanat, S. Doraivelu, P. Sadayappan, K. Schwan, and B. Weide. Parallel computers and finite element analysis. In *1986 ASME Int. Conf. Computers in Engineering*, pages 43–50, 1986.
- [5] R. Berrendorf. Memory access in shared virtual memory. In *Parallel Processing: CONPAR 92-VAPP V. Second Joint International Conference on Vector and Parallel Processing*, pages 785–786, 1992.
- [6] M. Berry and R. Plemmons. Parallel algorithms for finite element structural analysis on the HEP multiprocessor. In *Proc. Denelcor Workshop on the HEP*. University of Oklahoma, March 1985.
- [7] H. Blum. A transformation for extracting new descriptors of shape. In *Proc. Sym. Models for the Perception of Speech and Visual Form*, Cambridge, MA, 1964. MIT Press.
- [8] K. Boehncke, H. Heller, H. Grubmuller, and K. Schulten. Molecular dynamics simulations on a systolic ring of transputers. In A. S. Wagner, editor, *Transputer Research and Applications 3*, pages 83–94, Amsterdam, 1990. IOS Press.

- [9] Robert F. Boothe. *Evaluation of Multithreading and Caching in Large Shared Memory Parallel Computers*. Also tech report ucb-csd-93-766, University of California, Berkeley - Department of Computer Science, July 1993. 169 pages.
- [10] D. Brown, J. H. R. Clarke, M. Okuda, and T. Yamazaki. A domain decomposition parallelization strategy for molecular dynamics simulations on distributed memory machines. *Comput. Phys. Commun.*, 74:67–80, 1993.
- [11] D. Brown, J. H. R. Clarke, M. Okuda, and T. Yamazaki. A domain decomposition parallel processing algorithm for molecular dynamics simulations of polymers. *Comput. Phys. Commun.*, 83:1–13, 1994.
- [12] R.J. Brown and M.J. Manore. Gis use in analysis of remotely sensed imagery. In *Proceedings of the National Conference: Challenge for the 1990s - GIS*, pages 134–140. Canadian Institute of Surveying and Mapping, 1989.
- [13] J. Bruno. Final report on the feasibility of using the Massively Parallel Processor for large eddy simulations and other computational fluid dynamics applications. Technical Report 84.2, RIACS, June 1984.
- [14] M. Burger and F. G. Buchholz. On substructured finite element-and fracture analysis program for parallel processing on transputer networks in personal computers. In M. Valero, E. Onate, M. Jane, J. L. Larriba, and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 1280–1292, Amsterdam, 1992. IOS Press.
- [15] G. Carey. High speed processors and implications for algorithms and methods. In W. Wunderlich, E. Stein, and K. Bathe, editors, *Nonlinear Finite Element Analysis — Structural Mechanics*. Springer-Verlag, Berlin, 1981.
- [16] W. T. Jr. Carter. Finite element analysis on a transputer system. In D. L. Fielding, editor, *Transputer Research and Applications 4*, pages 88–96, Amsterdam, 1990. IOS Press.
- [17] T. Chan. On the implementation of kernel numerical algorithms for computational fluid dynamics on hypercubes. pages 747–755, 1987.
- [18] Ronald Chaney. Complexity as a sclae-space for the medial axis transform. Technical Report AI Memo 1397, January 1993.
- [19] C. M. Chen, S. Y. Lee, and Z. H. Cho. 3D PET image reconstruction on a mesh connected multiprocessor. In *Conference Record of the IEEE Nuclear Science Symposium and Medical Imaging Conference*, pages 915–17, Orlando, FL, 1992. IEEE.

- [20] Chung-Ming Chen. An inhomogeneous partitioning based parallel EM algorithm for 3D PET image reconstruction. *Biomedical Engineering, Applications Basis Communications*, 6(1):34–40, 1994.
- [21] Gang Cheng, Chris Faigle, Geoffrey Fox, Wojtek Furmanski, Bin Li, and Kim Mills. Exploring AVS for HPDC software integration: Case studies towards parallel support for GIS. Technical Report SCCS-473, Northeast Parallel Architectures Center, Syracuse University, 93.
- [22] Ming Chen Chung and Soo-Young Lee. On parallelizing the em algorithm for pet image reconstruction. *IEEE Transactions on Parallel and Distributed Systems*, 5(8):860–73, 1994.
- [23] S. Clar, B. Drossel, and F. Schwabl. Scaling laws and simulation results for the self-organized critical forest fire model. *Physical Review E*, pages 1009–1018, August 1994.
- [24] Keith C. Clarke, James A. Brass, and Philip J. Riggan. A cellular automaton model of wildfire propagation and extinction. *Photogrammetric Engineering and Remote Sensing*, 60(11):1355–1367, Nov. 1994.
- [25] T. A. DeFanti, M. D. Brown, and B. H. McCormick. Visualization: Expanding scientific and engineering research opportunities. *IEEE Computer*, 22(8):12–26. [8] 1989.
- [26] Andreas R. Dill, Martin D. Levine, and Peter B. Noble. Multiple resolution skeletons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):495–504, July 1987.
- [27] Q. Dinh, R. Glowinski, B. Mantel, J. Periaux, and P. Perrier. Subdomain solutions of nonlinear problems in fluid dynamics on parallel processors. In *5th International Symposium on Computational Methods in Applied Sciences and Engineering, Versailles, France*. North-Holland, 1981.
- [28] B. Drossel and F. Schwabl. Fomation of space-time structure in a forest-fire model. *Physica A*, 204:212–229, 1994.
- [29] L. C. Dutto, W. G. Habashi, M. Robichaud, and M. Fortin. Parallel strategy for the solution of the fully-coupled compressible Navier–Stokes equations. In M. N. Dhaubhadel, M. S. Engelman, and W. G. Habash, editors, *Advances in Finite Element Analysis in Fluid Dynamics American Society of Mechanical Engineers, Fluids Engineering Division*, volume 171, pages 21–31. ASME, New York. 1993.

- [30] C.R. Dyer and A. Rosenfeld. Thinning algorithms for gray-scale pictures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1(1), 1979.
- [31] Holt et al. An improved parallel thinning algorithm. *Communications of the ACM*. 30(2):156–160, Feb. 1987.
- [32] Hutchinson et al. Parallel neighbourhood modeling: Research summary. In *Proceedings of the Symposium on Parallel Algorithms and Applications*, pages 204–207, 1996.
- [33] James R. Beers et. al. Transputer performance issues using the trollius o/s. pages 35–49. IOS Press, 1990.
- [34] J.C. Eidenshink et al. Integration of remote sensing and gis technology to monitor fire danger in the northern great plains. In *Proceedings of the National Conference: Challenge for the 1990s - GIS*, pages 944–956. Canadian Institute of Surveying and Mapping, 1989.
- [35] Vipin Kumar et al. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. Benjamin/Cummings, first edition, 1994.
- [36] Thomas Fahringer, Matthew Haines, and Piyush Mehrotra. On the utility of threads for data parallel programming. July 1995. Also as Tech report 95-35 Institute for Computer Applications in Science and Engineering, NASA Langley Research Center.
- [37] G. D. Feitelson. *Optical Computing: A Survey for Computer Scientists*. MIT Press, first edition, 1988.
- [38] G. Fen Paw and D. R. J. Owen. Domain decomposition procedures for finite element analysis on transputer based parallel computers. *Comput. Syst. Eng.*, 2:451–460, 1991.
- [39] Afonso Ferreira and Nashib Qadri. Performance study of dynamic scheduling of in WTDM optical networks. 1995.
- [40] B. D. Fleisch and G. J. Popek. Mirage: A coherent distributed shared memory design. In *Proc. Twelfth ACM Symp. on Operating Sys., Operating Systems Review*, page 211, December 1989. Published as Proc. Twelfth ACM Symp. on Operating Sys.. Operating Systems Review, volume 23, number 5.
- [41] M. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computing*, C(21):94, 1972.

- [42] Foley, van Dam, Feiner, and Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 2nd edition, 1996.
- [43] T. J. Fountain. *Parallel Computing: Principles and Practice*. Cambridge University Press, University College, London, first edition, 1994.
- [44] R. Fulton. The impact of parallel computing on finite element computations. In *Reliability of Methods for Engineering Analysis*, pages 179–196. Pineridge Press, Swansea, 1986.
- [45] E. Gallopoulos. The Massively Parallel Processor for problems in fluid dynamics. In *Proc. Vector and Parallel Processors in Computational Science II Conference, Oxford, England, 1984*.
- [46] I. Gloudeman. The anticipated impact of supercomputers on finite element analysis. *Proc. IEEE*, 72:80–84, 1984.
- [47] David G. Green and Andrew Tridgell. Interactive simulation of bushfires in heterogeneous fuels. *Mathematical Computational Modelling*, 13(13):57–66, 1990.
- [48] T. Gross and A. Sussman. Mapping a single-assignment language onto the warp systolic array. In G. Kahn, editor, *Functional Programming Languages and Computer Architecture*, pages 347–363. Springer-Verlag, Berlin, DE, 1987. Lecture Notes in Computer Science 274; Proceedings of Conference held at Portland, OR.
- [49] Forestry Canada Fire Danger Group. Development and structur of the canadian forest fire behavior prediction system. Information report, Forestry Canada, Ottawa, Ontario, 1992.
- [50] Aarron Gull. *Cherub: A Hardware Distributed Single Shared Address Space Memory Architecture*. PhD thesis, Systems Architecture Research Centre, City University, March 1993.
- [51] Zicheng Guo and Richard W. Hall. Fast fully parallel thinning algorithms. *CVGIP: Image Understanding*, 55(3):317–328, May 1992.
- [52] John L. Gustafson, Gary R. Montry, and Robert E. Benner. Development of parallel methods for a 1024-processor hypercube. *SIAM Journal on Scientific and Statistical Computing*, 9(4), July 1988.
- [53] Richard W. Hall. Fast parallel thinning algorithms: Parallel speed and connectivity preservation. *Communications of the ACM*, 32(1):124–131, Jan. 1989.

- [54] P. Brinch Hansen. Parallel cellular automata: A model program for computational science. *Concurrency: Practice and Experience*, 5(5):425–448, Aug. 1993.
- [55] J. Hulskamp and D. Jones. Transputers and parallel applications. TAPA-92. proceedings of the 5th australian transputer and OCCAM user group conference. In *Amsterdam, Netherlands: Ios Press*, volume 221, pages 4–5. 1993.
- [56] Joji Iisaka and Takako Sakurai-Amano. Automated terrain feature detection from remotely sensed images integrating spectral, spatial and geometrical attributes of objects. In *GIS/LIS '95 Proceedings*, volume 1, pages 486–495, Nov 1995.
- [57] Joji Iisaka and Takako Sakurai-Amano. Lan-based integrated terrain understanding system for remote sensing. In *GIS/LIS '95 Proceedings*, volume 1, pages 496–504, Nov 1995.
- [58] Environmental Systems Research Institute. Arcstorm : The arc storage manager. White Paper, Feb. 1996.
- [59] Environmental Systems Research Institute. Spatial database engine (sde). White Paper, Mar. 1996.
- [60] Joseph JáJá. *An Introduction to Parallel Algorithms*. Addison Wesley, University of Maryland. first edition, 1992.
- [61] Kenneth Man kam Yip. KAM: Automatic planning and interpretation of numerical experiments using geometrical methods. Technical Report AI TR 1163, MIT. MIT, August 1989.
- [62] N. C. Kluksdahl, A. M. Krizan, and D. K. Ferry. The role of visualization in the simulation of quantum electronic transport in semiconductors. *IEEE Computer*, 22(8):60–67, [8] 1989.
- [63] P. E. Krueger, T.-H. Lai, and V. A. Radiya. Processor allocation vs. job scheduling on hypercube computers. In *Proceedings of the 11th International Conference on Distributed Computing Systems (ICDCS)*, pages 394–401, Arlington, TX USA, May 1991. IEEE Computer Society , Washington, DC.
- [64] Edward D. Lazowska. System support for high performance multiprocessing. In *Proceedings of the USENIX Symposium on Experiences with Distributed and Multiprocessor Systems*, pages 1–11, 1992.

- [65] Louis Leithold. *The Calculus of a single variable with analytic geometry*. Harper & Row, 5th edition, 1986.
- [66] G. Levi and U. Montanari. A gray weighted skeleton. *Inform. Contr.*, 17:62–91, 1970.
- [67] Bin Li. Opportunities and challenges of parallel processing in spatial data analysis : Initial experiments with data parallel map analysis. volume 2, pages 445–458, 1992.
- [68] G. K. Maa. Code-mapping policies for the tagged-token dataflow architecture. Technical Report MIT/LCS/TR-425, MIT Laboratory for Computer Science, May 1988.
- [69] David Martin. *Geographic Information Systems: Socioeconomic Applications*. Routledge, London, England, second edition, 1996.
- [70] V. Martorana, M. Migliore, and S. L. Fornilli. Parallel implementations of a molecular dynamics program for lennard-jones particles on transputer network. In T. Muntean, editor, *Parallel Programming of Transputer Based Machines*, pages 128–134, Amsterdam, 1988. IOS Press.
- [71] O. McBryan and E. Van de Velde. Hypercube algorithms for computational fluid dynamics. In *Proc. Oak Ridge Meeting on Hypercube Architectures, August 1985*. Society for Industrial and Applied Mathematics, to appear.
- [72] Maurice Milgram and Thierry de Saint Pierre. Boundary detection and skeletonization with a massively parallel architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(1):74–78, January 1990.
- [73] A. Nakano, R. K. Kalia, and P. Vashishta. Molecular dynamics simulation of aerogel silica on parallel computers. In *Materials Research Society Symposium Proceedings*, volume 293, pages 237–242. Pittsburgh, PA, 1993. Materials Research Society.
- [74] Heinrich Niemann. *Pattern Analysis and Understanding*. Springer Verlag, 2nd edition, 1990.
- [75] B. Nitzberg and V. Lo. Distributed shared memory: A survey of issues and algorithms. Technical Report CIS-TR-90-26, UNIV of Oregon, DEPT of Computer and Inf. Science, 1991.
- [76] Canadian Council of Forest Ministers. Compendium of canadian forestry statistics. Government publication, 1995.

- [77] S. Peleg and A. Rosenfeld. A min-max medial axis transformation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 3:208–210, 1981.
- [78] A. R. C. Raine. Molecular dynamics simulation of proteins on an array of transputers. In D. J. Pritchard and C. J. Scott, editors, *Applications of Transputers 2*, pages 272–279, Amsterdam, 1990. IOS Press.
- [79] G. Robinson. Parallel computational fluid dynamics on unstructured meshes. In M. Valero, E. Onate, M. Jane, J. L. Larriba, and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 1313–1322, Amsterdam, 1992. IOS Press.
- [80] L. J. Rosenblum. Scientific visualization at research laboratories. *IEEE Computer*, 22(8):68–101, [8] 1989.
- [81] David Roytenberg. Developing parallel gis applications on the alex avx ii parallel computer. Master's thesis, Carleton University, Ottawa, Canada, Sept. 1996.
- [82] D. Rutovitz. Pattern recognition. *J. Royal Stat. Soc.*, 129(Series A):504–530, 1966.
- [83] E. Salari and P. Siy. The ridge-seeking method for obtaining the skeleton of digital images. *IEEE Transactions on Systems, Man, and Cybernetics*, 14:524–528, 1984.
- [84] H. Schreiber, O. Steinhauser, and P. Schuster. Parallel molecular dynamics of biomolecules. *Parallel Computing*, 18(5):557–573, May 1992.
- [85] K. C. Sevcik. Application scheduling and processor allocation in multiprogrammed parallel processing systems. *Performance Evaluation North Holland*, 19, 2-3:107–140, 1994.
- [86] Horst D. Simon and Leonardo Dagum. Experience in Using SIMD and MIMD Parallelism for Computational Fluid Dynamics. Technical Report RNR-91-014, NASA — National Aeronautics and Space Administration, Ames Research Center, Moffett Field, California 94035, March 1991.
- [87] Michael J. Sullivan. Visualization: Computer graphics in the research laboratory. In *Fourth Computer Graphics Workshop Proceedings*, page 104, Cambridge, MA, October 8-9 1987. USENIX. Abstract only.
- [88] C. Dana Tomlin. *Geographic information systems and cartographic modeling*. Prentice Hall, Englewood Cliffs, New Jersey, 3rd edition, 1990.

- [89] R. Trobec, I. Jerebic, and D. Janezic. Parallel algorithm for molecular dynamics integration. *Parallel Computing*, 19(9):1029–1039, September 1993.
- [90] Y. F. Tsao and K. S. Fu. Stochastic skeleton modeling of objects. *Computer Vision, Graphics and Image Processing*, 25:348–370, 1984.
- [91] T.Y.Zhang and C.Y.Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, Mar. 1984.
- [92] Ronald C. Unrau, Orran Krieger, Benjamin Gamsa, and Michael Stumm. Experiences with locking in a NUMA multiprocessor operating system kernel. pages 139–152, 1994.
- [93] C.E. Van Wagner. Development and structure of the canadian fire weather index system. Technical report, Petawawa National Forest Institute, Chalk River, Ontario, 1987.
- [94] R. Watts. Spectral element methods for computational fluid dynamics on transputer arrays. In D. J. Pritchard and C. J. Scott, editors, *Applications of Transputers 2*, pages 472–479, Amsterdam, 1990. IOS Press.
- [95] R. Watts, J. S. Reeve, and O. R. Tutty. A portable parallel implementation of a domain decomposed computational fluid dynamics algorithm. In M. Valero, E. Onate, M. Jane, J. L. Larriba, and B. Suarez, editors, *Parallel Computing and Transputer Applications*, pages 1353–1362. Amsterdam, 1992. IOS Press.
- [96] Michael L. Wells and David E. McKinsey. Using a geographic information system for prescribed fire management at Cuyamaca Rancho State Park, California. In Michael Heit and Art Shortreid, editors, *GIS Applications in Natural Resources*, pages 337–342. GIS World Inc., Colorado, U.S.A., 1991.
- [97] Philip J. Woest and James R. Goodman. An analysis of synchronization mechanisms in shared-memory multiprocessors. Technical Report TR 1005, University of Wisconsin, February 1991.
- [98] B.M. Wotton. Great lakes forestry centre, personal communication, 1996.
- [99] B.M. Wotton and J. Renaud. A simple physical fire growth model. In *Proceedings of the 13th conference on fire and forest meteorology*, Lorne, Victoria, Australia, October 1996.
- [100] Yun Xia. Skeletonization via the realization of the fire front's propagation and extinction in digital binary shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(10):1076–1086. October 1989.

- [101] Jianping Xu and Richard G. Lathrop Jr. Improving cost-path tracing in a raster data format. *Computers and Geosciences*, 20(10):1455–1465, 1994.
- [102] W. S. Young and C. L. Brooks. Implementation of a data parallel, logical domain decomposition method for interparticle interactions in molecular dynamics of structured molecular fluids. *J. Comput. Chem.*, 15:44–53, 1994.
- [103] J. Zahorjan and C. McCann. Processor scheduling in shared memory multiprocessors. In *Proc. ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, page 214, May 1990.