1-1-1992

Parallel terrain feature extraction

Demetrius-Kl Rokos

Marc P Armstrong
University of Iowa

PARALLEL TERRAIN FEATURE EXTRACTION

Demetrius-Kl. Rokos
Marc P. Armstrong

Department of Geography
The University of Iowa
Iowa City, IA 52242

ABSTRACT

Spatial models often must handle large datasets to capture the complex interactions of physical systems. This leads to computational intensity. Fortunately, many geographical problems can be decomposed into relatively independent parts that can be processed simultaneously. This paper describes a terrain feature extraction algorithm that operates in a parallel computing environment. The morphology of a drainage basin is derived using information provided by a DEM. The time efficiency of the algorithm is investigated, as well as the relative speedup accomplished over a sequential version of the algorithm.

INTRODUCTION

As researchers have incorporated increasingly detailed and realistic descriptions of physical geographic processes in their models, data volumes and computational complexity have increased commensurately. The physical limit of processing speed (Polychronopoulos, 1988) and the cost of high performance traditional computers, however, have limited both the ability of researchers to engage in near-real-time modeling and the size of problems that can be examined. Parallel programming provides a solution to these constraints by integrating and harnessing the power of multiple processors to produce relatively inexpensive high performance computer environments. Before algorithms can be efficiently implemented in parallel architectures, however, they must be decomposed into relatively independent segments. Fortunately many spatial algorithms treat geographic space as an aggregation of interacting spatial units, thus facilitating their decomposition. This paper describes the decomposition and parallel implementation of a terrain feature extraction algorithm (Bennett and Armstrong, 1989). The algorithm is designed to operate on a digital elevation model (DEM) to determine the hydrologic category (e.g. channel, divide, slope break) of each DEM cell and then to inductively construct an initial approximation of the hydrologic feature networks.

FEATURE EXTRACTION ALGORITHMS

The detailed description and delineation of the subcatchments and hillslopes of a drainage basin are essential parts of many drainage basin studies. Because of their

well-defined boundary conditions and explicit internal connectivity, subcatchments and hillslopes provide an "unambiguous template for structuring models in geomorphology, hydrology, landscape ecology and other fields" (Band, 1989a:151). Two basic topographic features are used to derive this information: drainage and divide networks. Drainage networks provide a spatial framework for integrating spatial elements of the land surface and a structure for dissecting space into functional areas (Jarvis, 1977). The drainage network also defines the main flow pattern of water, sendiment, nutrients and pollutants in a watershed. The divide network, on the other hand, defines the extent and the shape of the runoff contributing area for each channel segment and acts as the boundary of relatively independent watershed partitions. As Band (1986) noted, encoding these feature networks may be quite tedious and time consuming for any but the smallest data sets which "has provided a strong restriction on the scale and the complexity of the watershed research that is generally attempted" (Band, 1989a:151). Researchers, therefore, have sought ways to automate this process.

Peucker and Douglas (1975) designed three feature extraction algorithms that are the basis of much of the consequent work in the area. The first used the elevation information in a 3 by 3 elevation window to classify the center cell of the window into categories (peak, pit, pass, ridge, ravine, slope, break and flat) using the following method:

1) Calculate elevation differences between the center DEM cell and its neighbors
2) Describe each feature type as a unique combination of the following characteristics: difference of the sum of all positive elevation differences with the sum of all negative elevation differences, number of sign changes and number of points between two sign changes.
3) Classify the center cell into a feature type by matching its characteristics with the characteristics of one of the prototypes created for the different feature types.

In the second algorithm the center cell is classified as a ridge if it is higher or a channel if it is lower than all four of its row and column neighbors. The third algorithm uses the basic principle of a "one-step move": In the neighborhood of every cell, the highest point in the window is flagged. All remaining cells are classified as channel cells. Similarly, if the lowest cell in every elevation window is marked, all remaining cells are classified as ridges.

In 1983 Mark proposed a process-based approach in which he assumes that "a drainage network represents those points at which runoff is sufficiently concentrated that fluvial processes dominate over slope processes" (Mark, 1983:289) and that the center cell in a 3 by 3 elevation window will drain to the neighbor having the steepest downslope towards it. Considering that every cell produces a unit quantity of runoff, every cell will drain this quantity plus the runoff of all the cells draining to it, if any. By using a predefined threshold over

which runoff could be considered concentrated, drainage network cells were defined as those cells that drain more water than the threshold. A more detailed implementation of the algorithm can be found in O'Callaghan and Mark (1984).

Marks *et al.* (1984) identified and delineated drainage basins and sub-basins using two measures: slope and exposure. Slope was defined as the angle between the terrain and the horizontal and exposure as the direction of slope. The goal of the algorithm is to identify which, if any, of the neighboring cells belong in the same basin as the center cell. A neighboring cell is in the same basin as the center cell, if its exposure faces toward the center cell within a quantization level of 8 divisions of the circle. Slope is considered only in almost flat terrain, when the exposure cannot be defined. A similar algorithm by Morris and Heerdegen (1988) also used drainage direction to derive the drainage network. Their algorithm went a step further since it used the drainage network to recursively define the boundaries of the drainage basin.

Jenson's (1984) algorithm assumes that if an elevation profile is V-shaped, then water will tend to concentrate there and form part of the drainage network. After identifying the drainage cells in a basin, the algorithm connects them using the following method: When two drainage cells are adjacent and the lower one belongs to a certain channel, then the other cell also belongs to the same channel. Finally, all non-drainage cells are assigned to the basin of a drainage network. This is accomplished by iteratively assigning labels to non-drainage cells that drain to (are uphill and adjacent to) already labeled cells.

In each of these algorithms, we described only the rules that are used to identify features. Following this step, each algorithm attempts to construct the topology of the feature networks by connecting adjacent cells that perform similar hydrologic functions. This step, however, often fails to produce the expected feature networks. Very often the original data sets include errors which appear in the form of pits or sinks, which rarely occur naturally (O'Callaghan and Mark, 1983). These artificial sinks cause the interruption or distortion of existing drainage lines or the introduction of erroneous non-existent drainage lines. There has been considerable discussion in the literature about techniques that either identify and remove erroneous pits from data sets (e.g. Marks *et al.*, 1984) or correct errors in the derived drainage lines (e.g. Band 1986; Morris and Heerdegen, 1988).

Another issue that feature extraction algorithms should address is scale. First, the degree of detail that an algorithm uses to represent feature networks should vary with respect to research objectives. Furthermore, there are no consistent criteria in the literature for the classification of a valley as a part of a drainage network or a ridge as part of a divide network. Since drainage networks are dynamically defined as the set of points where fluvial processes dominate over slope processes (Mark, 1983), their extent may be altered by changes in precipitation input and land cover. To deal with scale and dynamic feature definition issues, some feature extraction algorithms employ thresholds that enable the scale and the

detail of feature networks to be controlled. For example, Mark's (1983) algorithm classified a DEM cell as a drainage cell if the area contributing runoff to that point was larger than a predefined threshold.

In the next section we describe an alternative approach that addresses these shortcomings. This alternative, yet still computationally intensive, approach: 1) uses inductive logic to identify features and assemble them into networks, 2) allows control of the level of detail in feature network representation, and 3) can derive feature networks even in flat areas.

AN INDUCTIVE BIT-MAPPED CLASSIFICATION SCHEME FOR TERRAIN FEATURE EXTRACTION

The algorithm presented here uses a "rule based system driven by inductive logic and physical law" to extract hydrologically significant features from digital elevation models (Bennett and Armstrong, 1989:60). Induction encompasses "all inferential processes that expand knowledge in the face of uncertainty" (Holland *et al.*, 1989:1) and can be used when there are no objective and indisputable criteria to extract hydrologic features from digital elevation models. It should be noted that the results of the inductive process cannot be guaranteed to be correct. To improve the reliability of inductive predictions, domain specific knowledge is used and the predictions are fed back to the system for re-evaluation and improvement.

This algorithm was developed to extend previous work in 3 ways (Bennett and Armstrong, 1989:60): a) identify points at which the rate of flow is modified by relief (slope breaks), b) construct a topologically complete data structure that closely describes the topology of the basin, and c) subdivide the terrain into homogeneous areas with respect to slope and aspect. The algorithm consists of four steps:

1. The DEM cells are classified into hydrologic categories from which an initial approximation of the basin's morphology is derived.
2. The results are refined to improve the topographic model.
3. The fragmentation of the derived features is resolved by identifying and eliminating erroneous pits and connecting divides through passes.
4. The topological model of the basin is completed by connecting drainage and divide networks.

In this paper we will focus on the first step of this algorithm. In a 3 by 3 elevation window the algorithm records the two-dimensional shape of the four directed, symmetrical transects that cross the center cell. This information is used to classify each transect into one of the following hydrologic categories: 1) ridge, 2) valley, 3) slope break, or 4) flat. The shape of each transect is stored in a six element boolean array (Table 1) that is compared with prototypes generated for each hydrologic category. Each transect is placed in the category with which it

has the greatest similarity. This method was chosen because it has three advantages (Bennett and Armstrong, 1992):

1. Characteristics of features are captured in a simple manner avoiding redundancy.
2. Classification can proceed even if data are incomplete, erroneous, or inconclusive.
3. Classification is a simple pattern matching operation.

Bit Mapping:

1	true if center point is topographically significant
2	true if first point of transect is higher than mid point
3	true if last point of transect is higher than mid point
4	true if the transect is extended in the direction of the first point
5	true if the transect is extended in the direction of the first point
6	true if extension of either line reaches a boundary.

Line Classification:

ridge	Both bounding points lower than middle	#FF##F
valley	Both bounding points higher than middle	#TT##F
slope break	One bounding point higher, the other lower than the middle;	#TF### or #FT### or
	One end of the line extended, one not	###FT# or ###TF #
flat	Both ends of the line extended	###TTT

Table 1. Transect classification rules (from Bennett and Armstrong, 1989).

Next, topographic significance is used to define "the difference in elevation between the central point and a line bounded by two points on opposite sides of the window" (Bennett and Armstrong, 1989:62). The consideration of topographic significance in the transect classification serves two purposes. First, by using an elevation threshold below which a point is considered topographically insignificant the algorithm can account for noise in the digital elevation model that could cause erroneous classification of cells. Thus, the possibility of misclassification is reduced and classification is restricted to the points for which there is strongest evidence for their hydrologic function. Second, the topographic significance threshold can be used as a tool to control the degree of detail represented in feature networks.

The algorithm also can be applied to areas of low relief unlike many other feature extraction algorithms based in local operators (Band, 1986). Since in areas of low relief there may not be sufficient information for cell classification in a 3 by 3

elevation window, the algorithm allows for "flexible windowing". When the elevation of the center cell is not significantly different from either ends of a transect running through it, the window is recursively extended in the direction of uncertainty (Figure 1). The fact that a transect is extended beyond the 3 by 3 elevation window is recorded in elements 4 and 5 of the transect boolean arrays.

Using the information derived from the form of the four transects, the center cell of the elevation window is classified into one of the following six hydrologic categories: 1) divide point, 2) drainage point, 3) pit, 4) pass, 5) slope break, or 6) plain. After all cells have been placed in a hydrologic category, an initial approximation of the basin's morphology is derived by establishing preliminary topological relationships between classified cells. This is achieved by linking adjacent cells that perform similar hydrologic functions.

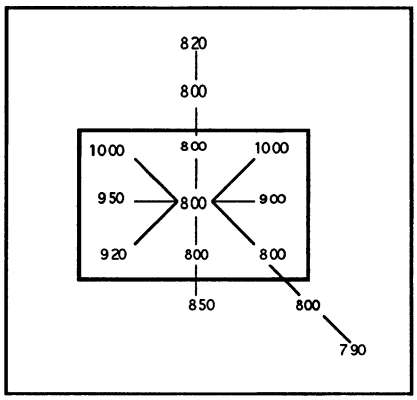

Figure 1. An example of transect extension in low relief areas (from Bennett and Armstrong, 1989).

PARALLEL PROCESSING

Parallel processing is a computing paradigm in which multiple computing resources are used to increase the amount of work performed per unit of time to solve a particular problem (Desrochers, 1987). Unlike sequential programming, however, an efficient implementation of a parallel algorithm is related to the architecture in which it is developed. To tailor an algorithm to the characteristics of the target architecture is a complex and tedious task (Polychronopoulos, 1988). Because of this complexity as well as the inadequacies of the software tools that are available for developing and debugging parallel programs (Mohiuddin, 1989), parallel processing has not been widely diffused. Despite these difficulties, parallel processing environments are now being developed by many computer manufacturers and will become increasingly widespread. Therefore, computationally intensive algorithms must adapt to the requirements of parallel processing.

Granularity of Parallelism

The decomposition of a problem into relatively independent tasks can be achieved in many ways. The grain size into which a problem is divided however, is fundamentally important. The granularity of an algorithm can be classified into three categories (Polychronopoulos, 1988):

- Fine grained algorithms have tasks equivalent to basic code blocks.
- Medium grained algorithms have individual tasks larger than a basic block but smaller than a subroutine.
- Coarse grained algorithms have tasks equivalent to one or more subroutines.

When deciding the most appropriate granularity for a problem two things should be considered in addition to what is the most natural decomposition strategy (Carriero and Gelernter, 1990: 14). First, the granularity of an algorithm is related to the granularity of the target architecture. "Fine grained" parallel computers with many single bit processors, for example, are not normally well-suited for coarse or medium grained parallelism. Second, it has been reported (Cok, 1991) that fine grained decomposition achieves a more efficient distribution of the workload than medium or coarse decomposition, but it usually requires a much greater communication overhead.

Parallel Programming Paradigms

Carriero and Gelernter (1991) introduced the term parallel paradigm to describe a distinct way of thinking about parallelism in an implementation independent context. There are three main parallel paradigms: event, geometric, and algorithmic parallelism. Two of these paradigms (event and geometric) are used to decompose the selected terrain feature extraction algorithm.

Event parallelism (also called the task or processor farm approach) involves the identification of a main process (master task or controller) and a set of specialized worker processes. The main task distributes data to worker processes and collects their results. The master task also schedules the work plan. Whenever the master task determines that a worker process is idle, it sends a new set of data. While event parallelism is inherently a load balancing paradigm, it incurs communication overhead as the controller must continuously send and receive data from the worker processes.

Geometric parallelism (also called data or result parallelism) involves the decomposition of the problem space into subregions within which local operations are performed (Healey and Desa, 1989). This is a popular paradigm as it "largely avoids the difficulty of finding a way to decompose a problem into parallel pieces" (Cok 1991). One very important issue that determines the efficient implementation of geometric parallelism in a particular problem is the interprocessor communication required. It is essential that the decomposition of

the problem space maximizes processing within a region and minimizes interregional exchange of information.

Process and Data Dependencies

When designing a parallel algorithm, the process and data dependencies of the problem domain must by studied to identify interactions and interdependencies between data entities. Problems that have complex process and data dependencies are difficult to decompose, which may lead to poor performance or even incorrect results if the correct sequence of execution is not guaranteed.

Data dependencies involve the interaction and interdependencies of data items on the instruction or loop level (Desrochers, 1987; Armstrong and Densham, 1992). Data dependencies can occur: a) during data input, b) when nested indexing is used to address array elements, and c) when elements of the same array are needed by different processes. Data dependencies can be generalized on the process or procedure level (Williams, 1990; Evans and Williams, 1980). Williams (1990) identified several different types of relationships that can exist between two processes (P1 and P2) in shared and private memory systems. Since the parallel architecture that has been used in this research is a distributed memory parallel computer we will briefly discuss the potential relationships between processes in a private memory system. The main types of relationships between processes are:

- **Prerequisite**: process P1 must fetch what it requires before P2 stores its results. P1 must not require results produced by P2, as there is no guarantee that P1 will finish before P2.
- **Conservative**: process P1 must send its results to a process P3 before P3 receives the results of P2. This suggests that both processes modify the same data structures and the results of process P2 are required for subsequent processing. This type of relationship allows the unconstrained parallel execution of the two processes until the time that the two processes have to return their results.
- **Commutative**: process P1 may execute before or after P2, but not at the same time. This occurs mainly when both processes modify the same memory locations during their execution.
- **Contemporary**: processes P1 and P2 are completely unrelated and may execute at the same time, without any constraints.
- **Consecutive**: process P1 must send all or part of its results to process P2. This implies that the results of P1 are used by P2 so the execution of the two processes is inherently sequential.

PROBLEM DECOMPOSITION

Before deciding on a decomposition strategy that would fit the characteristics of the selected algorithm, two things should be considered: the computer architecture and the process and data dependencies of the algorithm. Our programming

environment (a MIMD machine consisting of a 486 PC serially linked to 4 fully interconnected T800 Transputers) dictates that the problem domain should be decomposed into relatively large tasks (coarse grain parallelism) which require limited interprocessor communication. The data and process dependencies of the algorithm, on the other hand, complicate the decomposition process.

Dependency Analysis for the Feature Extraction Algorithm

In areas of low relief the determination of a transect's shape may require recursive extension. Since the length of the extension depends on local relief, it cannot be predefined. This causes inconveniences in distributed memory systems where each process can only access data from its own local memory. Also note that the relationship between different transect classifications is contemporary since each is independent and can execute in parallel. The information produced from the classification of the four transects (a bit-mapped boolean array) is used to classify the center cell of the elevation window. The type of relationship between these two processes (transect classification and cell classification) is consecutive, as the results of the classification of all four transects must be known prior to the classification of the center cell. The algorithm then connects adjacent cells that perform similar hydrologic functions. Within a DEM window the relationship between this connection step and the cell classification step is consecutive. However, when examined at a broader scale the two steps can be accomplished in parallel as the inter-window relationships are contemporary.

Because the algorithm exhibits different kinds of dependency relationships, several ways of organizing it for parallel processing can be devised. One way of decomposing the algorithm is into the three steps described earlier. This approach, however, creates relatively small tasks which require a large amount of communication (sending data and receiving results). Therefore, an alternative solution was developed to reduce overhead and improve performance. Since the first two steps of the algorithm, transect and cell classification, are closely related, and the results of the first are required by the second, they were combined together to form a single *cell classification* task. The third step of the algorithm, the *feature topology construction* task, is independent and executes after the cell classification has been completed. To coordinate the execution of these two worker tasks, a master task was also designed to distribute data to the worker tasks, receive their results and balance their workloads.

Problem Domain Decomposition

A simple way to decompose two-dimensional geographic data sets is geometric parallelism (Healy, 1989). In this approach the data set is divided in N equal parts, where N is the number of worker processors. Communication is required only for processing the boundary areas of each spatial subset. This approach works well for feature topology construction since simple data dependencies exist between the boundaries of the subsets. To process boundaries, boundary data between two adjacent subsets are made available to both of them and they are treated differently than interior cells. Furthermore, since each subset requires the

same amount of processing, if data are equally distributed among worker tasks, the workload of the worker processors is well-balanced and the computing resources are used efficiently.

The application of geometric parallelism in the first part of the algorithm (cell classification) presented serious problems. In flat areas, the transect may have to be extended to determine its shape. This requires that the elevation values in the direction of the extension are available to the cell classification task. Since the length of a transect's extension cannot be known *a priori*, the size of the elevation window that may be required for the classification of a single cell cannot be predefined, and therefore the master task must send the whole data set to each worker processor. Though this may be feasible for small data sets, it degrades the algorithm's performance for large data sets. Consequently, whenever a transect must be extended, the worker task encountering the problem sends all needed information to the master task, which has access to the entire data set. The master task extends the transect and returns its shape to the worker that requested it. Transect extension also dictates that if the geometric paradigm is used, performance problems may occur from uneven workload balancing. If a purely geometric decomposition were used, a worker processor receiving data representing a flat area would be required to perform many more transect extensions than other worker processors and would continue to process data after other workers have completed their tasks. To deal with this problem, the cell classification task was implemented using event parallelism. This paradigm inherently balances processing loads and therefore, if one worker is required to do more processing than others, it will execute fewer times.

PERFORMANCE ANALYSIS

Before attempting to evaluate our algorithm, several implementation issues closely related to the algorithm's performance must be addressed. First, the size of the individual workload in the cell classification task must be determined. Since our parallel architecture is used efficiently when the work within each processor is maximized and interprocessor communication is minimized, this might lead us to try to increase the individual workload size as much as possible. Such action, however, causes the same workload balancing problems as the application of the geometric parallelism paradigm. The ideal workload size should be a function not only of the data set size, but also of local relief. To provide a simple demonstration of the importance of workload size we ran the application with 3 x 3 and 3 x 31 workload sizes. Using the latter workload size we achieved a 28% performance improvement which was due to the increase of the unit computational step and the consequent decrease of the communication load.

The performance of the parallel implementation of the algorithm was also improved by reducing the length of the records sent to the worker tasks. Time was saved not only by assembling smaller records but also by reducing

communication overhead. To demonstrate this we executed the cell classification algorithm for the same data set using the old data structures of the sequential version and the new reduced ones. The results showed that the execution was 14.20% faster with the new data structures.

To evaluate other aspects of the performance of our parallel algorithm two measures were used: speedup and efficiency. **Speedup** (S) can be defined as the ratio of sequential run time (Tseq) to parallel run time (Tpar):

$$S = Tseq / Tpar \qquad (1)$$

Efficiency (E) can be defined as the ratio of the speedup (S) to the number of processors (N) running in parallel:

$$E = S / N \qquad (2)$$

We measured our algorithm's execution times when running it with one (sequential version), two and three worker processors (Table 2), then computed efficiency and speedup (Table 3). Figure 2 shows the graph of the execution time of the cell classification task and the feature topology construction task against the number of worker processors used. It is apparent from the timing results that by increasing the number of worker processors, a considerable speedup of the algorithm is achieved. In addition, Figure 2 indicates that by increasing the number of the worker processors, the relative performance improvement of the algorithm (efficiency) diminishes. Two factors explain this:

a) By increasing the number of the processors involved, more interprocessor communication is required.
b) The data set that has been used is quite small (27 X 31 points) and therefore the timing results are considerably influenced by communication overhead. If the program was executed with a larger data set the efficiency of the algorithm would not diminish as quickly.

Num. of Processors	Cell Classification	Feature Topology
1	9765	7390
2	5300	4700
3	3859	3526

Table 2. Timing measurements of the developed algorithm running the application with 1, 2 and 3 worker processors. Units are low priority Transputer clock tics (1 tic=64msec).

	Cell Classification		Feature Topology Construction	
Number of workers	Speedup	Efficiency	Speedup	Efficiency
2	1.84	0.92	1.57	0.79
3	2.53	0.84	2.10	0.70

Table 3. Measurements of the speedup and efficiency of the parallel version of the algorithm over its sequential counterpart, using 2 and 3 worker processors.

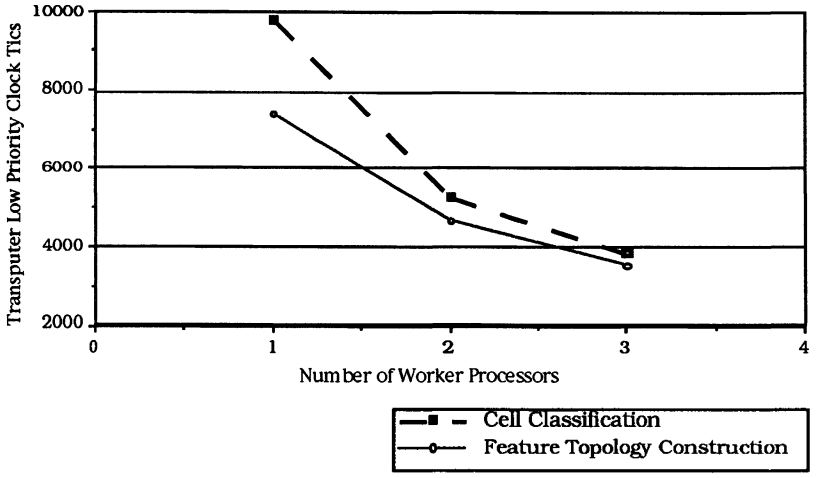

Figure 2. Graph of the algorithm's performance using 1, 2 and 3 worker processors

CONCLUSIONS

In this paper we briefly described the characteristics and principles of parallel processing that were applied to a terrain feature extraction algorithm implemented in a parallel environment. This application demonstrated the natural way that this particular problem can be decomposed into relatively independent tasks which are efficiently implemented in a parallel environment. The results showed the performance benefits of the parallel implementation over the traditional sequential approach. Also noted was the dependence of parallel algorithms on computer architectures, as a parallel algorithm has to be compatible with the processor granularity and the interprocessor communication topology of the target architecture. This hinders the wide application of parallel processing in many scientific fields such as geography where effort should be focused on analyzing the spatial interactions and dependencies of physical systems and processes to enable parallelism in an implementation-independent way.

ACKNOWLEDGMENTS

We wish to acknowledge comments and conversations with David Bennett and Paul Densham. Partial support for this research was provided by the National Science Foundation (SES-90-24278).

REFERENCES.

Armstrong M.P. and Densham P.J., 1992. Domain Decomposition for Parallel Processing of Spatial Problems. *Computers, Environment and Urban Systems* (forthcoming).

Band L.E., 1989. A Terrain-based Watershed Information System. *Hydrological Processes*, Vol. 3:151-162.

Bennett, D.A. and Armstrong, M.P. 1989. An Inductive Bit-Mapped Classification Scheme for Terrain Feature Extraction. *Proceedings GIS/LIS '89,* Orlando, FL.

Bennett, D.A. and Armstrong, M.P. 1992. A Knowledge-Based Approach to Topographic Feature Extraction.(unpublished manuscript)

Brawer S., 1989. *Introduction to Parallel Programming*. Academic Press Inc., San Diego, CA.

Carriero N. and Gelernter D., 1990. *How to write parallel programs. A first course*. The MIT Press, Cambridge, MA.

Cok Ronald S., 1991. *Parallel Programs for the Transputer*. Prentice Hall Inc., New York.

Desrochers G.R., 1987. *Principles of Parallel and Multiprocessing*. Intertext Publications, Inc., McGraw-Hill Book Company, New York.

Evans D.J. and Williams S., 1980. Analysis and Detection of Parallel Processable Code. *Computer Journal* 23:66-72.

Healy R.G. and Desa G.B., 1989. Transputer based Parallel Processing for GIS Analysis: Problems and Potentialities. *Proceedings of the Ninth International Symposium on Computer-Assisted Cartography, AUTO-CARTO 9*. Bethesda, MD: American Congress on Surveying and Mapping: 90-99.

Holland J.H., Holyoak K.J., Nisbett R.E. and Thagard P.R., 1989. *Induction: Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, MA.

Hopkins S. and Healey R.G., 1990. A Parallel Implementation of Franklin's Uniform Grid Technique for Line Intersection Detection on a Large Transputer Array. *Fourth International Symposium on Spatial Data Handling,* Zurich, Switzerland: 95-104.

Jarvis R.S., 1977. Drainage Network Analysis. *Progress in Physical Geography*, 1:271-295.

Jenson S.K., 1985. Automated Derivation of Hydrologic Basin Characteristics from Digital Elevation Model Data. *Proceedings of Auto-Carto 7*:301-310.

MacDonald R., 1989. Parallel Processing-Powerful Tool for the Earth Sciences. *U.S. Geological Survey Yearbook.*

Mark D.M., 1983. Automated Detection of Drainage Networks from Digital Elevation Models. *Proceedings of Auto-Carto 6*, Ottawa, Canada:288-298.

Marks D., Dozier J. and Frew J., 1984. Automated Basin Delineation from Digital Elevation Data. *Geoprocessing* 2:299-311.

Mohiuddin K.M., 1989. Opportunities and Constraints of Parallel Computing. In *Opportunities and Constaints of Parallel Computing*, Sanz G.L.C. (ed), Springer-Verlag New York Inc.

Morris D.G. and Heerdegen R.G., 1988. Automatically Derived Catchment Boundaries and Channel Networks and their Hydrological Applications. *Geomorphology* 1, Elsevier Science Publishers B.V., Amsterdam:131-141

O'Callaghan J.F. and Mark D.M., 1984. The Extraction of Drainage Networks from Digital Elevation Data. *Computer Vision, Graphics, and Image Processing* 28:323-344.

Peucker T.K. and Douglas D.H., 1975. Detection of Surface-Specific Points by Local Parallel Processing of Discrete Terrain Elevation Data. *Computer Graphics and Image Processing* 4:375-387.

Polychronopoulos C.D., 1988. *Parallel Programming and Compilers*. Kluwer Academic, Boston.

Williams S.A., 1990. *Programming Models for Parallel Systems*. John Wiley & Sons, Chichester, England.

3L Ltd, 1989. *Parallel Pascal User Guide*. 3L Ltd.