Improving the Performance of Raster GIS: A Comparison of Approaches to Parallelization of Cost Volume Algorithms.

Daniel F. Wagner
Michael S. Scott

Department of Geography
The University of South Carolina
Columbia SC 29208

phone: (803) 777-8976
fax: (803) 777-4972
{dan, mike}@lorax.geog.scarolina.edu

ABSTRACT
Performance evaluation and improvement are increasingly being recognized as critical elements for the success of Geographic Information Systems. This research examines one strategy for improving the efficiency of spatial algorithms: parallelization. The very nature of spatial data and operators is such that decomposition has obvious benefits. However, identifying the most beneficial approach for parallel decomposition of spatial operators is not obvious. This research examines the parallelization of a cost volume operator in raster space. Two distinct approaches to decomposition of the algorithm are undertaken on the 64-node MIMD Intel Paragon Supercomputer at the University of South Carolina. The different approaches are compared on several test workloads. Performance metrics include execution time, speed-up and efficiency. In addition, the relationship between data volume and the benefits of parallelization are considered.

1. Introduction

Performance evaluation and improvement are increasingly being recognized as critical elements for the success of Geographic Information Systems (GIS). As workloads continue to increase, performance constraints become more acute. Further, as system capabilities continue to expand, performance tends to actually *decrease* due to increased overhead of user interfaces, advanced spatial modeling capabilities, etc. Incremental advances in hardware technology can no longer be relied upon to provide the solution to GIS performance problems. Efforts must be made to increase the efficiency of spatial algorithms. Further, alternative hardware solutions must be considered.

As capabilities of GIS systems expand, for example true handling of three spatial dimensions, alternative platforms such as parallel processors will become requirements rather than options. Many spatial algorithms exhibit latent parallelism. This inherent parallelism is identified by Armstrong (1994) as one area of future work in the realm of GIS and High Performance Computing. Specifically, Armstrong asks the question "are there collections of code that can be translated simply into parallel versions - the so called embarrassingly parallel algorithms?" It is our assertion that the nature of many spatial algorithms are inherently parallel. Specifically, geometric strategies such as plane sweeping (Preparata and Shamos, 1985) in vector and roving windows in raster possess a high degree of parallelism. However, even among such 'embarrassingly parallel' algorithms a number of strategies are possible.

Algorithms are parallelized through decomposition. Decomposition may be carried out on the program code itself as well as on the problem space operated on. Further, both may be attempted in the same algorithm. No single strategy is best, and the considerations are many. Particularly acute is the number of processors available. In general, the more processors available the more options for decomposition, however, more processors only come at the expense of power or cost. In addition scalability must be considered. Scalability refers to how well an algorithm can be mapped to different numbers of processors. Algorithms which scale well show consistent improvements in performance as more processors are utilized. However, not all algorithms, nor all spatial algorithms scale well. Thus, the ideal system is not necessarily the one with the most processors.

This paper is arranged in several parts. This section has introduced the importance of parallel processing for overcoming the performance constraints of spatial algorithms, and has reviewed several applications of parallel processing of spatial algorithms. In the next section, the major issues in parallel processing are reviewed. Following this a brief survey of parallel processing of spatial algorithms is presented. The spatial algorithm which forms the focus of this research, 3-D raster cost volume generation is then introduced. Following this, the experimental design for the performance testing is presented. This is followed by the results of our preliminary analysis. Lastly, a discussion of these results and conclusions are provided.

2. Issues in Parallel Processing

2.1 Parallel Architectures

Two principle architectures for parallel processing machines exist. These are the Single Instruction Multiple Data stream (SIMD) and the Multiple Instruction Multiple Data stream (MIMD). In SIMD machines, all processors execute the same instructions in lock-step. Such systems, often referred to as array processors, are only suitable for a narrow set of applications. In MIMD machines each processor executes its own set of instructions. This may be the same set of instructions as other processors - but not executed synchronously - or processors may have different sets of instructions -in effect the processors do different jobs. MIMD machines may be further divided into shared and distributed memory systems. Shared memory has the effect of limiting the number of processors and performance is affected by competition for memory. In distributed memory systems, each processor has its own local memory and data is shared among processors through messaging. Such communication can result in processors being idle for long periods while waiting for data to be passed from other processors.

2.2 Decomposition Strategies

Parallelization strategies are commonly divided into control, domain, and hybrid.
In control decomposition, different functions are assigned to different processors. A common approach is pipe-line processing in which data flows from function to function and processor to processor sequentially. Another approach involves a manager-worker strategy in which one processor (the manager) assigns jobs to other processors (workers). In domain decomposition the data is partitioned among processors and each partition is assigned to a processor. Performance is largely a function of communication and balance and is affected by the granularity or size of the data partitions. Ideally, the partitioning strategy should conform to the problem/data. Since geographic problems are multi-dimensional, they provide a wealth of opportunities for partitioning.

2.3 Communication

Communication represents one potential bottleneck in parallel processing. Through message passing data can be shared across processors. Communication may take place either synchronously or asynchronously. For synchronous messaging both processors must be prepared. If the sending processor is not ready the receiving processor must wait. In asynchronous communication the receiver issues a request for data and can continue processing, however if the data has not been (fully) received by the time the receiver needs it problems will arise. Communication is related to partition size in that finer grain potentially requires more communication and may result in a poor communication/computation ratio. Some inherently parallel algorithms require little or no communication, while some inherently sequential algorithms will always run faster on sequential machines due to the communication bottleneck.

2.4 Balance

Balance refers to the distribution of workload among processors. If some processors are idle while others are active, poor balance exists. The total execution time for an application is equivalent to the execution time of the last processor to complete its task. If all processors complete at the same time, then execution time is optimized and perfect balance is achieved.

2.5 Performance

The most basic measure of system performance is execution time. A common strategy in benchmarking programs is to query the system clock both immediately before and immediately after a block of code. The difference between these two times is one measure of execution time (Wagner, 1991) . In parallel processing, comparisons between execution time for parallel programs running on a single processor (or sequential programs) and parallel multi-processor programs can be made. Speed-up (*S*) is the ratio between the time to execute on one processor (T_1) and the time to execute on *P* processors (T_P):

$$S = T_1 / T_P \tag{1}$$

and under ideal conditions:

$$T_P = T_1 / P \tag{2a}$$
$$S = P \tag{2b}$$

Efficiency is the ratio of speed-up to number of processors:

$$E = S / P \tag{3}$$

and under ideal conditions:

$$E = 1 \tag{4}$$

3. Parallel Processing of Spatial Algorithms

Parallel processing of spatial algorithms has been considered by a number of researchers. A variety of spatial operators have been the focus of parallel processing. These include line intersection detection (Franklin et al, 1989; Hopkins and Healey, 1990), polygon overlay (Hopkins and Waugh 1991), spatial statistics (Griffith, 1990; Rokos and Armstrong, 1993; Armstrong et al, 1994), location models (Armstrong and Densham, 1992); intervisibility and viewsheds (Sandhu and Marble, 1988; Mills et al, 1992), grid interpolation (Armstrong and Marciano, 1993), name placement (Mower, 1993a; 1993b), line simplification and point-in-polygon (Li, 1993), shortest paths (Sandhu and Marble, 1988; Ding and Densham 1992), and hill shading and drainage basin delineation (Mower 1993a).

Almost exclusively a data-parallel or domain decomposition approach has been taken to parallelization of spatial algorithms (Franklin et al, 1989; Hopkins and Healey, 1990; Hopkins and Waugh, 1991; Armstrong and Densham, 1992; Mills et al, 1992; Ding and Densham, 1992; Armstrong and Marciano, 1993; Mower, 1993a; Li, 1993; and Armstrong et al, 1994). Much less common are examples of a control-parallel or control decomposition approach to parallelization of spatial algorithms. Examples include: Mower (1993a) and Rokos and Armstrong (1993) who adopt a master-worker approach.

Parallelization has been undertaken on machines ranging from two (vector) processors (Sandhu and Marble, 1988) to massively parallel processors such as Connection Machines (Mills et al, 1992). Generally, however, the number of processors available has been small.

A number of insights have been noted by many sources: in general it can be observed that most spatial problems, and especially large ones, will benefit from parallelization, but that a variety of strategies, considerations, and liabilities exist. Of particular concern are communication bottlenecks (Mills et al, 1992), and the interdependencies of the particular problem (Rokos and Armstrong, 1993).

In addition to considering the approaches which have been undertaken, it is worthwhile to note that comparisons between control and domain decomposition have not been attempted for the same algorithm or on the same machine. In addition, hybrid decomposition has not been attempted for spatial algorithms. All of the reviewed applications have been restricted to 2-D (plane) or 2.5-D (surface) cases. Finally, while shortest paths have been considered by Sandhu and Marble (1988) and Ding and Densham (1993) only network (vector) paths have been considered, and best paths have not been considered at all. In this research we address all of these points through an attempt to examine and compare different strategies for parallelizing the same spatial algorithm. It is hoped that by comparing parallelization strategies, insights can be gained which can be extended to the parallel decomposition of other spatial algorithms.

4. Best Paths *in* 3-Dimensional Raster Space

Best path problems represent a generalization of the shortest path problem. In shortest path problems the path which minimizes distance between 2 points is determined. Best paths differ in that the minimization function is based not solely on distance but other criteria (e.g. time) as well. The effects of the other criteria are included in the form of impedances or frictions associated with movements in directions or over spaces. Best paths in 2-D raster space have been examined by a number of authors (Goodchild, 1977; van Oosterom, 1993; Wagner et al, 1993) however, extensions of the problem to 3-dimensions is relatively new (Scott, 1994a). Unlike shortest and best paths in network space which can take any direction, paths in raster space, may take only a fixed number of directions.

In this paper, we consider a 3-dimensional 26-direction raster cost volume operator which makes use of a 3-dimensional pushbroom. The algorithm was developed initially on a sequential processor (Scott, 1994a; 1994b) and resulted as a direct extension from an improved 2-D best path algorithm (Wagner et al, 1993). Pushbroom algorithms require multiple *sweeps* through the dataset (Eastman, 1989; Scott, 1994a) - one for each *broom* - and are thus computationally intensive.

The majority of the computation time in such best path algorithms is involved in the generation of the cost surface or volume which is subsequently used to generate the best path. If absolute barriers are allowed, a large number of *sweeps* may be required. In the 2-D case, four *sweeps* - one originating at each corner of the rectangular study area - are required (barrier free case), while in the 3-D case, eight *sweeps* are required - one from each of the eight corners of the rectangular prismoidal study volume (again, barrier free case). In both cases, each *broom sweeps* from a vertex to its diametrically opposed vertex. In the 3-D algorithm examined here, only 26 directions are utilized. In the 26-direction case, only the immediate neighborhood of a focus cell is considered. Such neighbors share a face, edge or vertex. Each of the eight *brooms* must consider 7 directions (Figure 1), thus some redundancy is unavoidable in the (sequential) algorithm. As example output, the x=25, y=10 planes of a 50x50x50 3-D cost volume (origin in center) are shown in Figure 2.

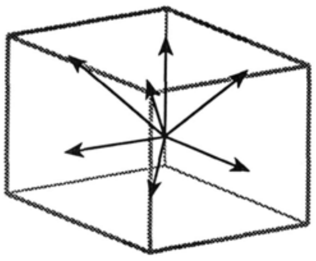

Figure 1. Seven directions considered by one *broom.*

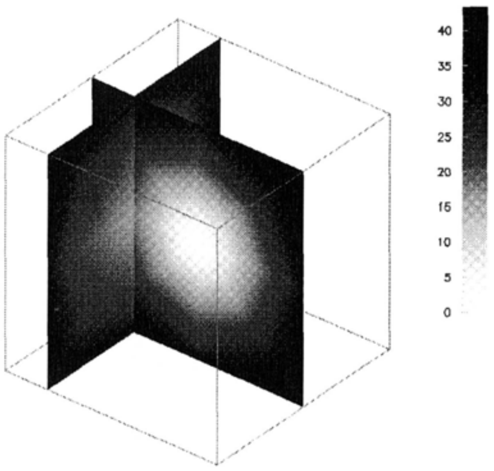

Figure 2. 50^3 Cost Volume, origin in center; planes: $x = 10, y = 25$

5. Experimental Design

5.1 Introduction

Tests were run on The University of South Carolina's Intel Paragon XP/S4 supercomputer. The Paragon has a distributed memory MIMD architecture using Intel 50 Mhz i860XP processors. The machine used in this research has 64 nodes arranged in a 2-D mesh, 1 gigabyte of distributed RAM, and a 12 gigabyte RAID. The nodes are divided between service and compute partitions. The service partition consists of 8 nodes each with 32 Megabytes RAM and handles logins, editing, compiling, utilities, and launching parallel programs. The compute partition contains the remaining 56 nodes, any number of which (including all) may be owned by a particular application. Each compute node has 16 megabytes of RAM and two i860XP processors, one for computing and the other for communication. Thus, 56 nodes represents the physical limit for this research. The i860XP processor is rated (peak) at 75 double precision megaflops computation and 200 MB/s communication. USC's Paragon has a peak performance of 4.2 gigaflops. All programming was done using C with parallel extensions.

The primary objective of this research is to examine and compare strategies for decomposition of spatial algorithms for parallel processing. We begin with the sequential algorithm, and then decompose this algorithm based on both control and domain. For each decomposition we examine the effect of scalability. Finally we examine a hybrid decomposition strategy which has elements of both control and domain decomposition.

Tests were performed on various sized synthetic datasets. The use of synthetic data allows for the total control needed to systematically examine the performance space of the system (Wagner, 1992). Results are reported in terms of execution time, efficiency, and speed-up.

5.2 Sequential Algorithm

The sequential algorithm can be conceptualized as a quintupley nested loop: For each of the eight *brooms*, and for each cell, the cost of moving from that cell to the neighboring cells is calculated and compared to the current cumulative cost for that neighboring cell and the minimum of the two is assigned. In pseudo-code, the algorithm may be envisioned as follows:

```
Broom
 Row
  Column
   Level
    Neighbor
     cmltv_cost(Neighbor)=
       MIN{cmltv_cst(Neighbor),cmltv_cst(Cell)+offset(Neighbor)}
     :
    :
   :
  :
 :
```

5.3 Parallel Algorithms

Our control decomposition is effected by parallelizing the outermost loop. That is the *brooms* sweep on several processors in parallel rather than each broom sweeping sequentially. In order to avoid the communications bottleneck only the calculation of cumulative cost is done within the loops and comparison and selection of minimum cumulative costs is delayed until a broom has completed its *sweep*. These values are then passed to a collector processor which determines the minimum value from the eight candidates assigned by the processors. Obviously in this approach anything beyond eight processors (or nine if the collector function is assigned to a separate processor) will not produce any performance improvements. For choices of 2, 4, and 8 processors, theoretically perfect parallelism can be achieved.

Control decomposition based on Neighbors rather than brooms is possible. Decomposition based on neighbors would involve assigning different directions to individual processors. All directions could be processed simultaneously (this would involve combining the outermost and innermost loops) using 26 processors. Collection would be required as above. This method of control decomposition is not implemented in this study.

Domain decomposition requires division of the study volume into sub-volumes. A major consideration for push-broom algorithms is that the broom must be able to sweep from corner to corner. In essence, the global effect must be preserved, so that a phenomenon can propagate throughout the entire study volume. For the cost volume algorithm, preserving the global effect requires multiple passes to allow effects to propagate across the sub-domain boundaries.

Sub-volumes are assigned to individual processors and the (sequential) algorithm applied. In order to allow propagation across sub-domain boundaries with a minimum of communication, the concept of 'ghost' cells (Intel 1994) is adopted. Rather than making the domain decomposition a mutually exclusive and collectively exhaustive one, the sub-volumes are designed to overlap. For pushbroom algorithms, the required depth of overlap is one cell. These overlapping cells are termed ghost cells. The ghost cells facilitate communication between passes and reduce the communication bottleneck.

Theoretically, perfect parallelism can be achieved if the sub-volumes are all the same size and shape. Perfect balance is achieved when the decomposition is uniform in all three directions. Balance is a function of the number of passes which must be performed to preserve the global effect. The number of passes required is equal to the maximum number of subdivisions of the problem volume along any axis. That is if the problem volume is bisected along an axis 2 passes are required; if the problem volume is trisected along an axis, 3 passes are required, etc. In effect, perfect balance is obtained when each spatial dimension is decomposed by the same integer (the cube root of the number of sub-volumes). Thus perfect balance and perfect parallelism can be achieved for 8, 27, 64, ... sub-volumes.

Hybrid decomposition, in which both control and domain decomposition are combined, is possible. Such a scenario might involve decomposing the study volume into eight sub-volumes and decomposing the sequential algorithm as described above. In this way better scalability should be achievable. That is, larger numbers of processors can be incorporated effectively. Hybrid decomposition is not attempted in this study.

6. Results

Control decomposition utilizing a collector and between 1 and 8 *broom* processors (2-9 processors) was performed using cubic study volumes ranging from 10x10x10 to 100x100x100 cells. Domain decomposition utilizing 1, 2, 8, and 27 sub-domains was performed on cubic study volumes ranging in size from 10x10x10 to 100x100x100. Results of individual tests were performed several times and averaged in order to lessen residual background effects. The results from the tests are presented in Tables 1-3.

Scott (1994a) developed the original 3-dimensional 26-direction raster cost volume operator in Turbo Pascal on a (sequential) PC-based environment. He reports that using a 33 Mhz 486DX processor with 16 megabytes of RAM of which 8 megabytes were configured as a RAM disk, approximately 1069 seconds were required to generate a 40x40x40 cost volume, and 79,000 seconds were required to generate a 100x100x100 cost volume. He also notes that the performance of his algorithm is very much disk constrained. Our first step in this research was to port Scott's sequential algorithm to C and a RISC/UNIX environment. On a SPARC IPX, the algorithm required 21 seconds to generate the 40x40x40 cost volume and 1200 seconds to generate the 100x100x100 cost volume. Next, the sequential algorithm was run on the Paragon (Table 1). Following this we undertook control and domain decomposition of the algorithm. Tables 2-3 present the results of these experiments.

Table 1. Sequential Program: Execution time in seconds.

Number of Nodes	Problem Size									
	10^3	20^3	30^3	40^3	50^3	60^3	70^3	80^3	90^3	100^3
1	0.10	0.98	3.49	8.50	16.9	29.4	47.0	70.6	102.	139.

The sequential tests show almost constant linearity between test size and execution time. The performance of the Paragon in sequential mode appears an order of magnitude better than the IPX for the 100^3 case, and two orders of magnitude better than the PC. Tables 2-3 illustrate that the sequential times are slightly faster than the parallel times using one node. This is due to the overhead associated with the parallel processing. This disadvantage vanishes as multiple-nodes are used, as expected. In the 100^3 tests, the sequential processor again demonstrates the best performance due to overhead associated with problem size in parallel mode.

Table 2. Control Decomposition: Execution time in seconds.

Number of Nodes	Problem Size									
	10^3	20^3	30^3	40^3	50^3	60^3	70^3	80^3	90^3	100^3
1	0.12	1.02	3.57	8.69	17.2	30.0	47.9	71.9	118.	225.
2	0.06	0.57	1.97	4.76	9.48	16.5	26.4	39.6	66.2	193.
3	0.11	0.47	1.59	3.77	7.44	13.0	20.7	30.7	45.0	238.
4	0.08	0.37	1.25	2.95	5.81	10.1	16.1	23.5	35.1	258.
5	0.23	0.53	1.37	3.11	6.06	10.4	16.3	24.7	35.9	315.
6	0.32	0.56	1.37	2.94	5.56	9.62	15.0	22.7	33.0	341.
7	0.33	0.58	1.30	2.83	5.45	9.61	15.2	23.0	31.9	401.
8	0.39	0.54	1.30	2.73	5.14	8.74	13.8	20.5	29.2	457.

Table 3. Domain Decomposition: Execution time in seconds.

Number of Nodes	Problem Size									
	10^3	20^3	30^3	40^3	50^3	60^3	70^3	80^3	90^3	100^3
1	0.24	2.01	7.08	17.2	34.1	59.4	94.9	142.	215.	374.
2	0.16	1.15	3.93	9.42	18.6	32.3	51.4	77.0	223.	429.
8	0.42	0.92	2.41	5.16	9.74	16.8	26.1	39.2	57.7	455.
27	0.86	2.39	6.43	14.1	26.5	45.0	70.7	105.	157.	1268

6. Discussion

The observed execution times illustrate that the 3-D 26-direction raster cost volume generation algorithm does benefit from parallelization. However, the observations also suggest that the algorithm does not scale evenly and that data volume can have a significant impact on speed-up and efficiency. In general the observations for the smallest problems (10x10x10) are least interesting. This is because the execution times are so small as to be completely confounded by the background processing overhead.

The control decomposition strategy works consistently better than the domain decomposition strategy on this algorithm. Closer examination of the parallel algorithms suggests that this is due largely to the communication required in the domain decomposition. In control decomposition, communication is necessary only at the beginning and ending of the processing. At the beginning, the origin(s) and friction values must be passed to each processor. Upon completion of the *sweep* the cumulative cost values must be passed to the collector processor where the minimum value is selected. Thus, little communication is needed, and what communication there is does not interfere with the individual *broom* processing.

In comparison, the domain decomposition strategy involves much more communication and further, this communication occurs during the sub-domain processing. While attempts have been made to minimize communication through the use of ghost cells, communication between all sub-domains must occur between each pass, and presents a significant bottleneck. The execution times for domain decomposition are on average nearly twice as long when compared to control decomposition.

Another observation centers on performance versus problem size. In both the control and domain tests anomalies in the pattern of performance can be observed as the problem size grows large. Visual inspection of the processor status has shown that these performance drop-offs occur when the workload becomes disk intensive. Problems up to some threshold size can be handled entirely in RAM, but beyond this threshold, disk storage must be utilized. Once this threshold is passed, all processors spend a majority of their time waiting for the RAID to service their requests for data. In addition, performance degradation occurs in the preprocessing stage when large numbers of processors or large amounts of array space (i.e. large data sets) are utilized. Again, visual observation of processor status has shown that while processors and/or memory is being allocated, other processors generally sit idle.

In addition to raw measure of execution time, speed-up and efficiency should be considered. Tables 4-7 present the speed-up and efficiency measures for both experiments. In general, speed-ups increase moderately as processors are added. However, this rate is clearly not proportional to the change in number of processors. Thus, while the greatest speed-ups are gained using the most processors, the greatest efficiencies are gained using the smallest number of processors. In addition, as execution times increase, speed-up and efficiency both drop-off with very large problems or large numbers of nodes.

A final topic for discussion is balance. Near perfect balance is possible in both the control and domain strategies outlined here. In fact, in certain instances the control decomposition for the cost volume algorithm represents the ideal case. While the processor workloads can be equally well balanced in the domain strategy, it is the communication between processors which offsets this advantage.

Imperfect balance occurs in the domain decomposition strategy if the sub-domains are not all the same size and shape. That is balance will be upset if some processors must operate on larger sub-domains. Imperfect balance occurs in the control decomposition case when the number of *broom* processors is not a factor of eight. Thus, examination of Table 2 shows that there is little difference in performance times for 4, 5, 6, and 7 processors. In effect, with more than three but less than eight processors, two passes are still required to process all eight *brooms*. Thus, some of the additional processors sit idle while others are busy on the second pass. In other words, such numbers of processors throw the parallelization greatly out of balance.

Table 4. Speed-Up from Control Decomposition.

Number of Nodes	Problem Size									
	10^3	20^3	30^3	40^3	50^3	60^3	70^3	80^3	90^3	100^3
2	2.00	1.79	1.81	1.83	1.81	1.82	1.81	1.82	1.78	1.16
3	1.09	2.17	2.25	2.31	2.31	2.31	2.31	2.34	2.62	0.95
4	1.50	2.76	2.86	2.95	2.96	2.97	2.98	3.06	3.36	0.87
8	0.31	1.89	2.75	3.18	3.35	3.43	3.47	3.51	4.04	0.49

Table 5. Efficiency from Control Decomposition.

Number of Nodes	Problem Size									
	10^3	20^3	30^3	40^3	50^3	60^3	70^3	80^3	90^3	100^3
2	1.0	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.6
3	0.4	0.7	0.8	0.8	0.8	0.8	0.8	0.8	0.9	0.3
4	0.4	0.7	0.7	0.7	0.7	0.7	0.7	0.8	0.8	0.2
8	0.0	0.2	0.3	0.4	0.4	0.4	0.4	0.4	0.5	0.1

Table 6. Speed-up from Domain Decomposition.

Number of Nodes	Problem Size									
	10^3	20^3	30^3	40^3	50^3	60^3	70^3	80^3	90^3	100^3
2	1.50	1.75	1.80	1.83	1.83	1.84	1.85	1.84	0.96	0.87
8	0.57	2.18	2.94	3.33	3.50	3.54	3.64	3.62	3.73	0.82
27	0.28	0.84	1.10	1.22	1.29	1.32	1.34	1.35	1.37	0.29

Table 7. Efficiency from Domain Decomposition.

Number of Nodes	Problem Size									
	10^3	20^3	30^3	40^3	50^3	60^3	70^3	80^3	90^3	100^3
2	0.8	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.5	0.4
8	0.1	0.3	0.4	0.4	0.4	0.4	0.5	0.5	0.5	0.1
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.0

7. Conclusions

We have examined two distinct strategies for parallelization of 3-D raster cost surface generation. In this test, control decomposition appears to hold significant advantages over domain decomposition. While domain decomposition offers better scaling and can be spread over a broader range and greater number of processors, the increased communication necessary for this approach ultimately lessens its performance.

While we are confident our control decomposition is optimal, we are aware of a number of potential strategies for improving our domain decomposition. This research has shown that control decomposition can be superior to domain decomposition for spatial problems. However, additional research is necessary before generalizations can be drawn. As in many other studies, we conclude that the most successful strategy is the one making use of both the greatest number of balanced processors and the least amount of communication. In the example of 3-D raster cost volumes, near perfect balance and near perfect parallelism can be exploited in the control decomposition. While (near) perfect balance can be easily achieved in our domain decomposition, the degree of communication necessary to preserve the global effect and allow values to propagate across sub-domains insures that near perfect parallelism (i.e. little communication) is not possible.

Additional strategies for decomposing this algorithm are also worthy of investigation: these include control based on the 26-directions and hybrid decomposition strategies. In light of the communications bottleneck associated with domain decomposition, hybrid strategies should prove a viable alternative to facilitate increasing the number of processors utilized while maintaining satisfactory perforce. In order to achieve this, the maximum number of *brooms* should be included since this 'control' component of the hybrid decomposition is not so communication bound.

In addition to examining alternate strategies, it is necessary to investigate the effect of architecture on the success of the decomposition strategies. Finally, it is necessary to more rigorously examine the performance space to better determine the existence and extent of efficiency thresholds or discontinuities in the performance space.

Acknowledgements
The authors would like to thank Dr. Patrick Fay, computational scientist at USC's Computer Services Division, for his invaluable assistance with the Paragon.

Bibliography

Armstrong, M., and Densham, P. 1992. "Domain Decomposition for Parallel Processing of Spatial Problems." *Computers, Environment, and Urban Systems.* Vol 16, pp 497-513.

Armstrong, M., and Marciano, R. 1993. "Parallel Spatial Interpolation." *Proceedings*, Auto-Carto 11. Bethesda, MD: ASPRS and ACSM. pp 414-423.

Armstrong, M. 1994. "GIS and High Performance Computing." *Proceedings*, GIS/LIS '94. Bethesda, MD: ASPRS and ACSM; Washington, DC: AAG and URISA; Aurora CO: AM/FM International. pp 621-630.

Armstrong, M., Pavlik, C., and Marciano, R. 1994. "Parallel Processing of Spatial Statistics." *Computers & Geosciences.* Vol 20, pp 91-104.

Eastman, J. R. 1989. "Pushbroom Algorithms for Calculating Distances in Raster Grids." *Proceedings*, Auto-Carto 9. Bethesda, MD: ASPRS and ACSM. pp 288-297.

Ding, Y., and Densham, P. 1992. "Parallel Processing for Network Analysis: Decomposing Shortest Path Algorithms for MIMD Computers." *Proceedings*, Spatial Data Handling 5. Columbia, SC: Department of Geography, University of South Carolina, IGU Commission on GIS. pp 682-691.

Franklin, W. R., Narayanaswami, C., Kankanhalli, M., Sun, D., and Wu, P. 1989. "Uniform Grids: A Technique for Intersection Detection on Serial and Parallel Machines." *Proceedings*, Auto-Carto 9. Falls Church, VA: ASPRS and ACSM. pp 100-109.

Goodchild, M. 1977. "An Evaluation of Lattice Solutions to the Problem of Corridor Location." *Environment and Planning A.* Vol 9, pp 727-738.

Griffith, D. 1990. "Supercomputing and Spatial Statistics: A Reconnaissance." *The Professional Geographer*, Vol 42, pp 481-492.

Hopkins, S., and Healey, R. 1990. 'A Parallel Implementation of Franklin's Uniform Grid Technique for Line Intersection Detection on a Large Transputer Array." *Proceedings*, Spatial Data Handling 4. Columbus, OH: Department of Geography, Ohio State University, IGU Commission on GIS. pp 95-104.

Hopkins, S., and Waugh, T. 1991. "An Algorithm for Polygon Overlay Using Cooperative Parallel Processing." *Working Paper No. 19.* Edinburgh: Economic and Social Science Research Council Regional Research Laboratory for Scotland.

Intel Supercomputer Systems Division. 1994. "Paragon™ User's Guide." Beaverton: OR.

Li, B. 1993. "Suitability of Topological Data Structures for Data Parallel Operations in Computer Cartography." *Proceedings*, Auto-Carto 11. Bethesda, MD: ASPRS and ACSM. pp 434-443.

Mills, K., Fox, G., and Heimbach, R. 1992. "Implementing an Intervisibility Analysis Model on a Parallel Computing System." *Computers & Geosciences.* Vol 18, pp 1047-1054.

Mower, J. 1993a. "Implementing GIS Procedures on Parallel Computers: A Case Study." *Proceedings*, Auto-Carto 11. Bethesda, MD: ASPRS and ACSM. pp 424-433.

Mower, J. 1993b. "Automated Feature and Name Placement on Parallel Computers." *Cartography and Geographic Information Systems.* Vol 20, pp 69-82.

van Oosterom, P. 1993. "Vector vs. Raster-based Algorithms for Cross Country Movement Planning." *Proceedings*, Auto-Carto 11. Bethesda, MD: ASPRS and ACSM. pp 304-317.

Preparata, F., and Shamos, M. 1985. *Computational Geometry: An Introduction.* New York, NY: Springer-Verlag.

Rokos, D., and Armstrong, M. 1993. "Computing Spatial Autocorrelation in Parallel: A MIMD implementation for Polygon Data." *Proceedings*, GIS/LIS '93. Bethesda, MD: ASPRS and ACSM; Washington, DC: AAG and URISA; Aurora CO: AM/FM International. pp 621-630.

Sandhu, J., and Marble, D. 1988. "An Investigation into the Utility of the Cray X-MP Supercomputer for Handling Spatial Data." *Proceedings*, Spatial Data Handling 3. Columbus, OH: Department of Geography, Ohio State University, IGU Commission on GIS. pp 253-267.

Scott, M. 1994a. "Optimal Path Algorithms in Three-Dimensional Raster Space." Unpublished M.S. Thesis. Columbia, SC: Department of Geography, University of South Carolina.

---------. 1994b. "The Development of an Optimal Path Algorithm in Three-Dimensional Raster Space." *Proceedings*, GIS/LIS '94. Bethesda, MD: ASPRS and ACSM; Washington, DC: AAG and URISA; Aurora CO: AM/FM International. pp 687-696.

Wagner, D. 1991. "Development and Proof-of-Concept of A Comprehensive Performance Evaluation Methodology for Geographic Information Systems." Unpublished Ph.D. Dissertation. Columbus, OH: Department of Geography, Ohio State University.

-------------. 1992. "Synthetic Test Design for Systematic Evaluation of Geographic Information Systems Performance." *Proceedings*, Spatial Data Handling 5. Columbia, SC: Department of Geography, University of South Carolina, IGU Commission of GIS. pp 166-177.

Wagner, D., Scott, M., Posey, A. 1993. "Improving Best Path Solutions in Raster GIS." *Proceedings*, GIS/LIS '93. Bethesda, MD: ASPRS and ACSM; Washington, DC: AAG and URISA; Aurora CO: AM/FM International. pp 718-726.