WORKING PAPER 96/2

Modelling and Optimising Flows Using Parallel Spatial Interaction Models

Ian Turton and
Stan Openshaw

Published March 1996

Abstract

The paper demonstrates some of the benefits that high performance computing has to offer geographers. It reports the results of porting and then using very large spatial interaction models on the Cray T3D parallel supercomputer in modelling and spatial optimisation applications that would otherwise have been judged computationally infeasible.

Keywords optimisation, simulated annealing, parallel design, data parallel, retail networks, modelling

Background

The entropy-maximising spatial interaction model is used for modelling many different types of flow data in both research and applied contexts. Examples of flow data are: journey to work trips, airline traffic, retail behaviour, world trade and telephone traffic in which flows of people, money, information etc., connect origin zones to destination zones. These zones have a geographical expression on maps whilst the flow data provides a useful summary of the very complex processes and behaviour patterns that generated them; for example, journey to work flows summarise the spatial dimensions of labour markets covering home-workplace interactions whilst in a retail context credit card purchases connect people's home addresses to where they shop. Not surprisingly building computer models of flow data has many commercial applications. The purpose of this paper is to describe a few applications as a means of illustrating some of the benefits that high performance computing (HPC) can offer computationally minded geographers interested in this area of human systems modelling.

Flow data and spatial interaction models

The idea of developing a parallel spatial interaction model is not new. Harris (1986), discussed how (in theory) it could be ported onto a parallel machine. Openshaw (1987) presented some further ideas regarding vector supercomputers. However, it seems that not until the early 1990s was the parallel implementation of the spatial interaction model developed, see Birkin et al (1995). The reasons for wanting to parallelise spatial interaction models are: to allow much bigger datasets to be processed more speedily than previously, to improve the underlying science both by being able to use finer resolution data, and to seek an improvement in the quality of the results by using more flexible methods of parameter estimation and model optimisation: In the spatial interaction model the compute load is an approximate power function of the number of zones, small numbers of zones run well enough on a PC but large numbers (i.e. few thousand) require high performance computing hardware in the form of vector or parallel supercomputers. An ability to model large flow data matrices is now important because the availability and volume of flow data has rapidly increased as a result of developments in IT, viz. data warehousing and there is a prospect of UK flow data tables with up to 1.6 million origins and destinations becoming available soon (viz. credit card transactions) or 32 million origin and destinations (if telephone traffic were to be modelled at the subscriber level). However, currently in the UK the biggest public domain flow matrices are those from the special workplace statistics of the 1991 census (Openshaw, 1995). This is a 10,764 by 10,764 matrix of journey to work flows between all wards in Britain.

A simple origin constrained flow model developed using entropy-maximising methods is specified as follows:

$$T_{ij} = O_i D_j A_i e^{-\hat{\beta} C_{ij}} \qquad (1)$$

$$A_i = 1 / \sum_j D_j e^{-\hat{\beta} C_{ij}} \qquad (2)$$

where T_{ij} is the predicted number of trips from an origin place i to a destination place j, O_i is the "size" of i, D_j is the "size" of j, and C_{ij} is a measure of the distance or travel cost between i and j. The parameter $\hat{\beta}$ is estimated to optimise the fit of the model using maximum likelihood or nonlinear least squares methods. Note that equation (2) ensures that the predicted flows satisfy the following accounting constraint:

$$\sum_j T_{ij} = O_i \qquad (3)$$

A doubly constrained model is slightly more complex:

$$T_{ij} = O_i D_j A_i B_j e^{-\hat{\beta} C_{ij}} \qquad (4)$$

$$A_i = 1 / \sum_j D_j B_j e^{-\hat{\beta} Cij} \qquad (5)$$

$$B_j = 1 / \sum_i O_i A_i e^{-\hat{\beta} C_{ij}} \qquad (6)$$

This model has two sets of accounting constraints: equation (3) and

$$\sum_i T_{ij} = D_j$$

This is the purpose of the A_i and B_j terms, which incidentally are functions of each other and have to be estimated iteratively.

Porting and scalability experiments

Fortran code was written for the two models so that they could cope with the 10,764 origin and destination zones of the 1991 ward level journey to work data. The two matrices used in the model to hold observed trips and costs (T_{ij} observed, C_{ij}), could not be stored in the memory of a standard UNIX workstation, since they would require over one Gbyte of memory. A solution to this is to store the trip matrix as a singly linked list which reduces the storage needed to about five Mbyte but this requires that the cost matrix can be recomputed when needed rather than stored. Unlike the trip data, the cost matrix is dense and cannot be stored in sparse format. Repeatedly recalculating the C_{ij} values as they are needed dramatically reduces the memory requirements, however, this is achieved at the expense of a large increase in the amount of computation. On a Sunsparc 10/41 workstation the singly constrained one parameter model took 17.6 hours to run. The non linear optimisation routine performed 29 model evaluations in the search for an optimal parameter. The doubly constrained version ran for 264 hours. It is clearly difficult to make much use of a model that runs for 11 days before a single result is obtained. The question now is how much faster would a parallel version run and what new applications a dramatic speed-up in model performance might provide.

Initially the code was ported to the Kendall Square Research KSR1 parallel super computer at Manchester University (Openshaw and Sumner, 1995). This is a virtual shared memory multiprocessor with 64 processing units. It is an MIMD architecture with a strongly coherent, global shared memory, which is implemented across its physically distributed processor memories by a combination of operating system software and hardware. Each processor is a 20 Mhz super-scalar RISC chip with a peak 64 bit floating point performance of 40 Mflop/s and 32 Mbyte of local memory. Therefore the KSR1-64 at Manchester has a theoretical peak performance of 2.5 Gflop/s and 2 Gbyte of memory. The codes for both models are easily parallelised by adding compiler directives to the serial code. Of course this data parallel approach will only work well if the code begin ported is already implicitly suitable for concurrent computation. The spatial interaction model is an example of what might be regarded as an embarrassing parallel model that is well suited to both vector and parallel supercomputers without much algorithmic change. The principle areas of parallelisation in the model are in the vector computation of the A_i and B_j terms and running the final model, in that the model's predictions for each of the destination zones are independent, they can be assigned to many different processors and thus performed concurrently.

The code was also ported onto the Cray T3D parallel supercomputer at Edinburgh. The Cray T3D version runs on 256 DEC alpha processors each rated at 150 Mflop/s with 64 Mbytes of memory, providing a peak theoretical speed of 38.4 Gflop/s for the whole machine. The same data parallel strategy was used for the KSR. In both cases the parallelism was at the outer DO LOOP level, so that in effect the computational load of the model was split into 10764 concurrent pieces, one for each i value. This is a fairly fine grained level of within model parallelisation but was the highest level relevant to this model in a parameter estimation context. Here the model subroutine was going to be called 50 to 100 times by a nonlinear optimiser as it searches for an optimal parameter values and for this application there is no benefit in seeking to parallelise the nonlinear optimisation algorithm itself.

Figure 1 shows the performance for the origin constrained model (equation 1) with comparisons being made to the KSR1. As can be seen it scales extremely well and whilst hardly surprising this is very reassuring. Of more interest is the discovery for this model that the Cray T3D is running only about twice as fast as the KSR1 for a comparable number of processors. This is less than was expected (the T3D processor speeds are almost 4 times faster than the KSRs) and seems to reflect the T3D's restriction that shared memory arrays on the T3D are a power of two. In this context the nearest power of two to the 10,764 zones is 16,384, which results in 34% of the processors being idle if block memory allocation is used and about 20% for the banded allocation used here. This can be regarded as a load balancing problem which is not caused by the model (since the computational loading for each *i* is about the same) but by the architecture of the Cray machine.

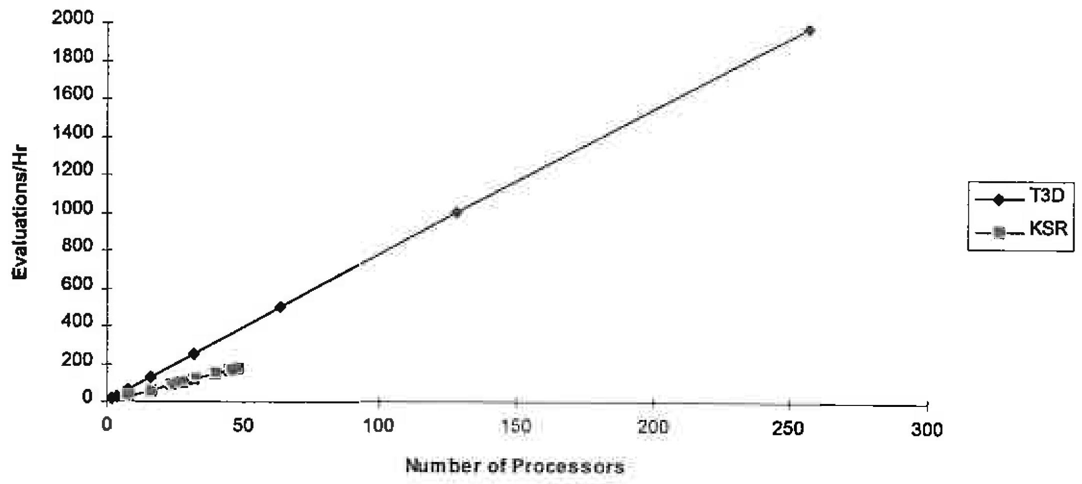

Figure 1: Speed-up of the origin constarained model

The results in figure 2 are of greater potential interest. This shows the performance of the doubly constrained model (equation 3) also scales in a linear fashion. Again performance scales in an almost linear fashion. It is interesting to note that **one** model evaluation on a SunSparc 10/41 workstation took 91 hours of compute time compared to 171 seconds on the 256 processor T3D. The equivalent times for the simpler origin constrained model are 36 minutes on the Sun and 1.8 seconds on the T3D. It is now possible for the first time to easily model journey to work patterns for the whole of Britain at the finest available level of geographic data resolution. Furthermore, the compute times are now sufficiently small to allow newer and more complex types of models to be investigated which are more compute intensive and more advanced forms of parameter estimation procedures to be applied; for example using genetic algorithms to estimate the $\hat{\beta}$ values which require about two to three orders of magnitude increase in computation but which may often yield better results; (Diplock and Openshaw 1996).

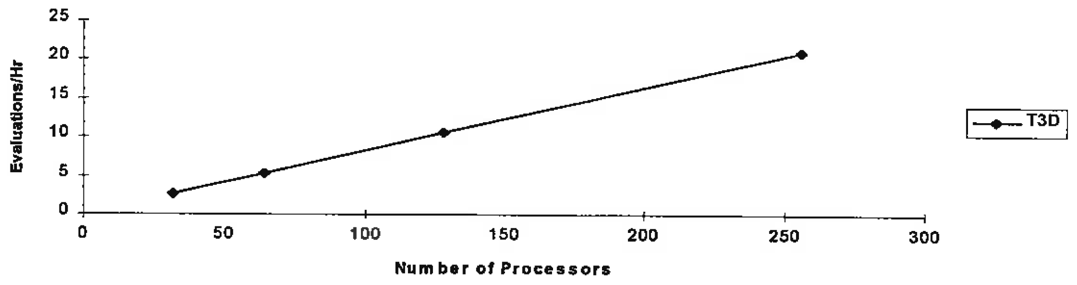

Figure 2: Speedup of the doubly constrained model

Parameter surfaces

Flow data is an immensely rich information source. The 1991 journey to work data for Britain provides a snap shot of the entire national space economy. Once there is an ability to model such large flow matrices then interesting new research opportunities arise. Of particular interest here is the extent to which each of the 10,764 destination zones may have associated with them a different $\hat{\beta}$ value (see equation 1). This involves running 10,764 spatial interaction models, each time estimating a $\hat{\beta}$ value for each destination zone. In effect each row or column of the matrix is processed as a separate run. The results are shown in Figure 3. They pick out rural-urban differences in the deterrent effect of distance on journey to work flows. The peaks corresponding to local labour markets and shows an un-precedent amount of spatial detail.

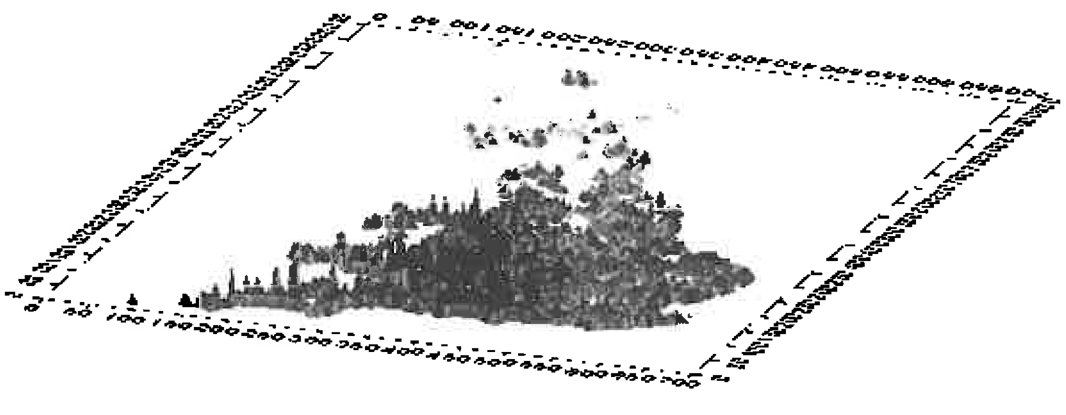

Figure 3: Beta values for journey to work flows at ward level for Great Britain

Retail network optimisation

Another major applied use for these models is to embed them in a spatial optimisation framework (Birkin, et al., (1995). For example in a retail context or hospital planning situation, it is often of interest to determine which set of D_j values (in equation 1) will yield maximum profits or optimise some other performance indicator of global network benefits. This involves solving a discrete nonlinear combinatorial programming problem using various computationally intensive heuristics; e.g. genetic algorithms, tabu search and simulated annealing. The quality of the results now depends heavily on, put crudely, the number of times a model such as that in equation 1 can be evaluated in a fixed time period on a realistically sized spatial interaction dataset.

Consider the problem of optimising a national retail network using data for 2755 consumer origin zones and 822 destinations. The aim is to solve a massive combinatorial optimisation problem that involves finding the best 60 sites from 822 alternative locations that maximise profits assuming consumers behave according to the spatial interaction model. There are 822!/60! possible solutions. This spatial network optimisation model can be written as:

$$\text{maximise} \quad \sum_{i}^{2755} \sum_{j}^{822} T_{ij}(\hat{D}_j^{\,1}) \tag{7}$$

$$T_{ij} = O_i D_j^{\,1} A_i e^{-\hat{\beta} C_{ij}} \tag{8}$$

$$A_i = 1 / \sum_{k=1}^{2} \sum_{j}^{822} D_j^k e^{-\hat{\beta} C_{ij}} \tag{9}$$

The aim is to define a set of 0, 1 values for variable D_j^1 such that the objective function is optimised and $\sum_{j=1}^{822} D_j^1 = 60$. The D_j^2 values represent existing competitor sites and are fixed. Obviously different sets of D_j^1 will alter the total market share attracted to these locations. Other constraints on the distribution of D_j^1 values may also be imposed; for example, minimum distances between nearest non-zero D_j^1 values should exceed a distance threshold.

When the serial code for this model was parallelised at the *i* loop level the performance no longer scaled with the number of processors (see figure 4). This problem was smaller than previously and the code was also heavily optimised with the result that the amount of work being performed in the parallel regions rapidly became less than the communication overheads. As the number of processors increased, the performance peaked (at 16 processors) and then got worse. The problem can be temporarily fixed for the T3D by increasing the amount of work being done (i.e. removing some of the optimisation) but this defeats the objective of maximising the number of model evaluations per hour. Another solution was to make the entire model a parallel activity and not just the major loop. This was possible because of the nature of the optimisation task in particular the simulated annealing process used to optimise equation (7) was based on a single move heuristic that used a local neighbourhood search. The search neighbourhood could be of any size and by setting this to some integer multiple of the number of processors being used then high levels of parallel efficiency and linear scalability were achieved; see Table 1. This leads to a peak performance of 28.2 MFlops/PE which is good for a Cray T3D.

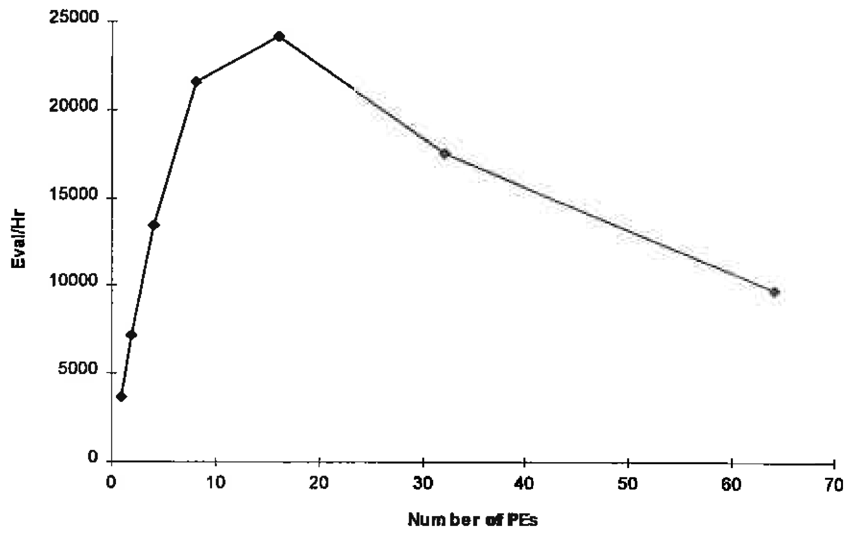

Figure 4: Speedup of the spatial optimisation program.

Table 1 also shows the impact of various forms of tuning on this code. It emphasises the important point that here the benefits of parallelisation were less than the effects of the serial code optimisations that were performed in attempts to optimise performance on a parallel machine. For example, the serial code optimisation yielded a speed up factor of 1097 which could have been attained on a workstation. This was achieved by unrolling do loops and cache optimisation (by hand), by noting that in equation (9) the k=2 loop was a constant result and could be stored (version 1 in Table 1) and also that equations (7), (8), and (9) could be computed in a sparse form, thereby greatly reducing the amount of arithmetic being performed without losing generality (version 2 in Table 1). This destroys the vectorisable nature of the model but puts it into a much more efficient form for a parallel machine. Running it on a 256 processor T3D yielded another factor of 256. The final combined speedup compared with the serial code (running on hardware 10 times slower than an Alpha chip) was an astonishing 2.8 million times equivalent to 70 million model evaluations per hour of T3D time. This might be compared with a speed up factor of 2260 reported by Birkin et al (1995) on a Thinking Machine CM-200. As well as a speed up the new algorithm also produced nearly twice as good a result as the serial version; see Figures 5 and 6.

Table 1: **Improvements in speed due to code optimisation and parallelisation**

Code	Time on 1 PE (Seconds)	Model Evaluations/Hr	Models Evals/ Hr on 256 PE	Improve-ment
Original serial Code	14.260	252.45	64628.3	
Memory Used to reduce computation	1.300	2769.23	708923.1	10
Hand Optimisation	0.690	5217.39	1335652.2	20
Version 1	0.037	97297.30	24908108.1	385
Version 2	0.025	144000.00	36864000.0	570
Version 2 + Blas	0.013	276923.08	70892307.7	1096

Our experiences to date suggest that parallel programming is fairly easy in principle but extremely time consuming to obtain the best possible levels of performance on a particular machine architecture. It would have been considerably easier and much quicker in terms of effort to have simply stopped after parallelising the original code with memory used to reduce compute times, perhaps forgoing the additional factor of 1000 speedup by running on a T3D compared with a workstation. Whether the additional computation is worthwhile depends on the commercial (or research) need to find a near optimal rather than merely a very good result to equation (7); and the extent to which the computing costs are affordable.

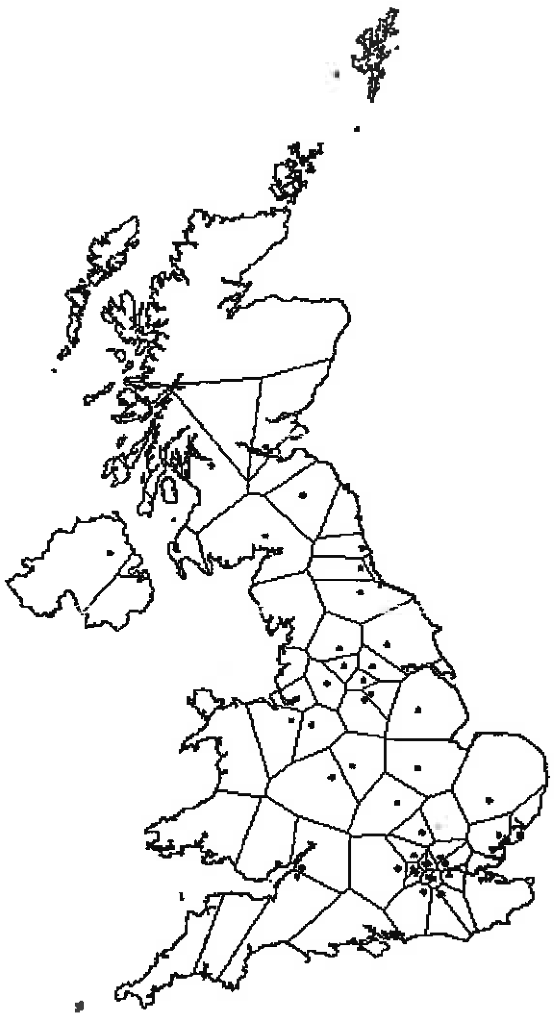

Figure 5: The optimal dealer network found with the serial code

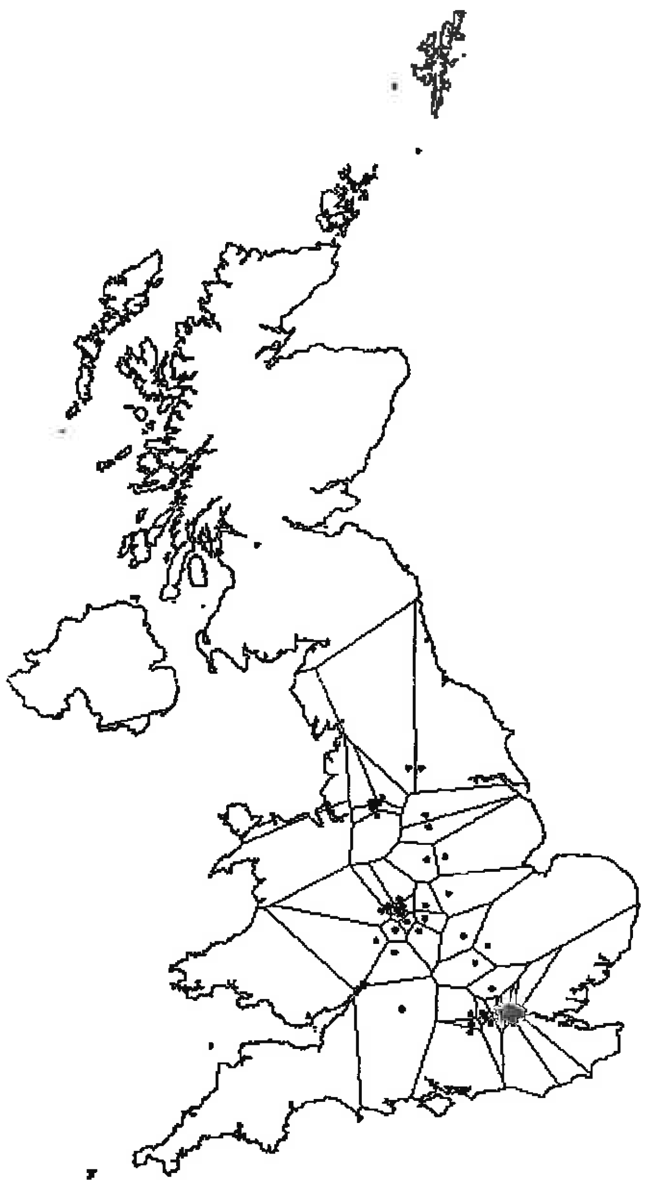

Figure 6: The optimal dealer network found using the improved algorithm on the Cray T3D

Conclusions

The paper describes the development of a parallel spatial interaction model and some applications involving two of the largest flow matrix yet modelled. The results are interesting both in their own right and also as a means of increasing awareness of what high performance computing (HPC) in general, and parallel super computing in particular, can offer. Increasingly geographical modellers are being freed from the computational constraints that have existed ever since the early years of the quantitative revolution. As Openshaw (1994b) points out, this could be the dawn of a major new era in geography, the beginning of what can be termed "computational geography"; defined as new computationally intensive ways of doing geography. HPC is not just a means of making old models run two or three orders of magnitude faster but much more significantly it creates an opportunity to solve many previously unsolvable problems, to develop new computationally based technologies and to create entirely new ways of thinking about and doing all kinds of geography not all of which were previously quantitative. Hopefully the results reported here will help stimulate awareness and assist in this process of exploiting HPC in geography and the social sciences.

Acknowledgement

This research was funded by the EPSRC (Grant number GR/K43933).

References

Birkin, M., Clarke, M., and George F. (1995), 'The use of parallel computers to solve nonlinear spatial optimisation problems: an application to network planning', *Environment and Planning A, 27, 1049-1068*

Diplock, G., Openshaw, S., 1996 'Using simple Genetic Algorithms to calibrate Spatial Interaction Models', *Geographical Analysis*, (forthcoming)

Harris, B., (1985), 'Some notes on parallel computing with special reference to transportation and land use modelling', *Environment and Planning A*, 17, 1275-1278

Openshaw, S., (1987), 'Some applications of supercomputers in urban and regional analysis and modelling', *Environment and Planning A*, 19, 853-860

Openshaw, S., (1994a) 'Some geographical applications of high performance computing', *Working Paper 94/18,* School of Geography, Leeds University

Openshaw, S., (1994b) 'Computational Human Geography: towards a research agenda', *Environment and Planning A*, 26, 499-505

Openshaw, S., 1995, Census Users Handbook GeoInformation International, Cambridge

Openshaw, S. and Sumner, R., 1995, 'Parallel spatial interaction modelling on the KSR1-64 supercomputer', Working Paper 95/15, School of Geography, University of Leeds