Some Geographical Applications of Genetic Programming on the Cray T3D Supercomputer

I. Turton, S. Openshaw and G. Diplock
School of Geography, University of Leeds, Leeds, UK
email: ian@geog.leeds.ac.uk, stan@geog.leeds.ac.uk, gary@geog.leeds.ac.uk

April 15, 1996

Abstract

The paper describes some geographical applications of a parallel GP code which is run on a Cray T3D 512 processor supercomputer to create new types of well performing mathematical models. A series of results are described which allude to the potential power of the method for which there are many practical applications in spatial data rich environments where there are no suitable existing models and no soundly based theoretical framework on which to base them.

1 Introduction

The Geographical Information Systems (GIS) revolution together with the computerisation of management and administrative systems has produced large amounts of geographically referenced information covering many aspects of modern life. The challenge now is how best to use these databases to create new knowledge in areas where current theoretical understanding is weak, how to develop or discover new process models of the behaviour of the human systems that the databases reflect and how to invent new and more appropriate exploratory analysis systems. This paper deals with the task of developing new process models of geographical systems able to cope with the complex properties of spatial phenomena. This task is becoming increasingly urgent because of the difficulties of building geographical models by more traditional methods. Additionally, many of the existing models are old and predate the spatial data explosion that has occurred since the late 1980s.

Traditionally, mathematical models are specified on the basis of good or strong theoretical knowledge but in many social sciences the available theories are suspect and at best poor. Additionally, the geographical systems of interest are usually highly complex, non-linear, probably chaotic and currently are not fully or properly understood due to the immense complexity of the human and environmental systems that are involved. One way forward is to use artificial neural networks and fuzzy logic modelling as universal approximators in the hope that viable computer models can be obtained by these essentially black box methods ; see Openshaw (1992, 1996). These inductive approaches use machine learning techniques to create equation free representations that 'learn' how to map a set of inputs to one or more outputs and in the process provide good fits to observed data. However, these methods are far removed from the traditional equation based mathematical model building activities that geographers and the other social scientists have used.

The paper explores an alternative model building approach based on the application of Genetic Programming methods run on a Cray T3D with 512 processors. This has the advantage of retaining the algebraic symbolism of the traditional approach whilst removing the total dependency of the model design and model discovery process on the skills of the human being. Instead high performance computing is used to build new computer models using what might be termed a generic type of model breeding machine. Section 2 outlines the design of model breeders whilst section 3 discusses their implementation on the Cray T3D parallel supercomputer. The results of some of the early applications are presented in Section 4.

2 Early Model Breeding Machines

The idea of building models by computer is not new. For over 30 years certain classes of linear statistical models have been created by examining all permutations of the predictor variables; for instance, if there are M predictors there are 2^{M-1} possible linear regression models than can be built, explored, and the best one identified. There is no reason why a similar strategy cannot be used to build mathematical models except once the model form is no longer restricted to being linear the number of possible permutations becomes far too large to examine in an exhaustive manner. There are also other problems in that the parameters now have to be estimated using a non-linear optimisation procedure and this may well involve a three or four orders of magnitude increase in compute times for a single model.

The success of computer automated model design now critically depends on developing an efficient search process. For instance, how can you be reasonably confident that by creating and evaluating 10^4 models you have found the best or near best models from a search space that may well contain more possible model equations? Equally even if there exists an old model how can you be reasonably sure that this model is the best (or amongst the best) model that exists. One approach is to generate and evaluate as many randomly generated model equations as possible in a fixed period of compute time; see Openshaw (1986). This model crunching strategy would at least allow current conventionally produced models to be viewed in terms of a broader context. However, whilst surprisingly good levels of performance can often be achieved a much better approach is to base the model generation process on some kind of intelligent search or optimisation process. Early attempts involved the use of focused Monte Carlo search methods and simulated annealing (Openshaw, 1988).

It soon became apparent that it would be in principle far better to use a genetic algorithm to drive this model search process. This is investigated in Openshaw (1988) who developed what was termed an Automated Modelling

System (AMS). A basic genetic algorithm (BA) based on Holland (1976) was used to breed simple mathematical equation based models. The entire process was powered by various Cray IS and XMP supercomputers in the mid 1980's. These machine generated equation based models were evaluated in terms of their ability to fit a dataset. The problems were two-fold: (1) insufficiently powerful supercomputers and (2) difficulties in representing models as bit strings.

The basic idea is still relevant and it makes much sense to regard many model design problems as a search problem. In many modelling applications it is quite apparent that the available model pieces (variables, parameters, unary mathematical functions, binary operators, and rules of arithmetic equations) can be combined in many different ways to form an immense universe containing all possible model equations that could be built. In the original research the principal problem was how best to represent symbolic equations based on the range of available model pieces by a fixed length bit string that would nevertheless allow the GA maximum freedom to create and search the universe of all potentially possible model equations for good performing models. Two different representational schemes were used. The first assumed a very simple structure as shown in Figure 1. Note that each equation could have a number of unknown parameters and these were estimated using a non-linear least squares procedure which was embedded in the GA. However, the basic Darwinian, rather than Lamarckian, philosophy was retained and the optimal parameters were not stored as part of the bit string but merely used in computing the fitness function.

A more complex alternative coding scheme was developed whereby the bit string was decoded to form a reverse polish expression of the model equation. This was subsequently used in a commercial version developed from AMS system called OMIGA (Barrow 1993) . Figure 2 outlines this model representation. A model consists of a number of these genes. The genes are sorted by their status (providing position independence) and then used to create a model in a reverse polish form. The model equation that emerges is the longest complete equation that satisfies the reverse polish logic implicit in the ordering of the model pieces. The problem is that this type of implementation is hard for the GA to handle, due to the high probability of redundancy and unused bits, its variable length and its self defining nature. However, it does work, although there is a feeling that improved results might be obtainable if a better way of representing the problem could be found.

3 Genetic Programming based Model Breeders

3.1 Serial GP

In this paper we attempt to improve on AMS by using the newer technique of genetic programming (Koza, 1992). Figure 3 outlines the basic structure of a Genetic Programming approach. GP is basically a GA that applies modified genetic operators not to bit strings but directly to the symbolic equations. This method has the outstanding advantage that the problem representation is much

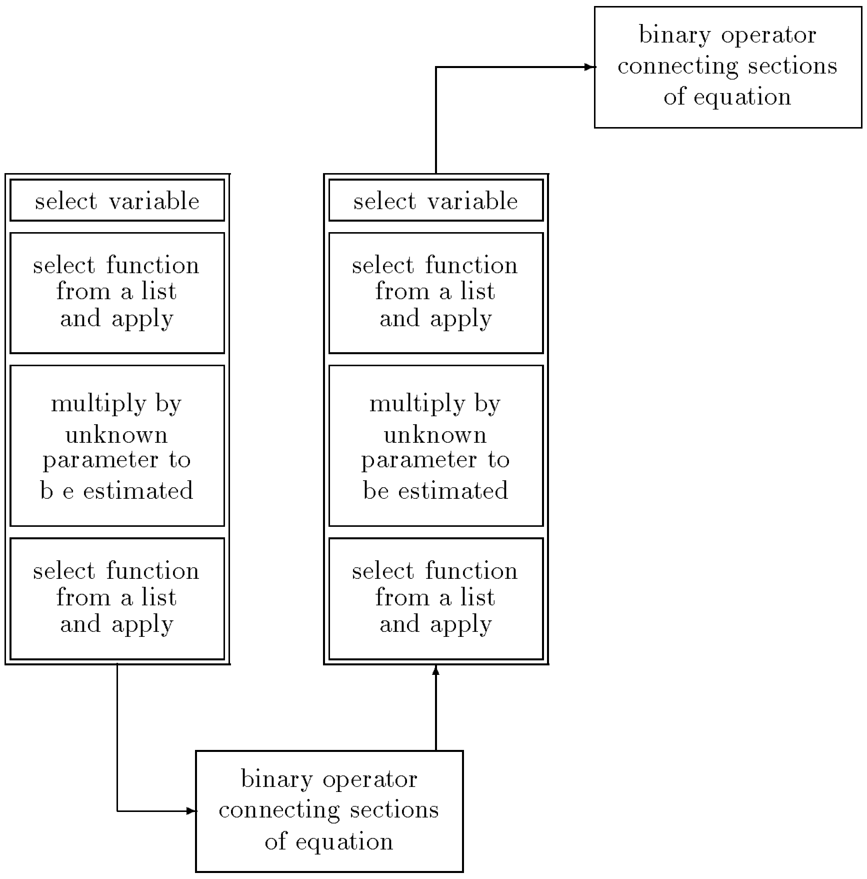

binary operator connecting sections of equation
select variable
select function from a list and apply
multiply by unknown parameter to b e estimated
select function from a list and apply
select variable
select function from a list and apply
multiply by unknown parameter to be estimated
select function from a list and apply
binary operator connecting sections of equation

Figure 1: A simple model representation scheme

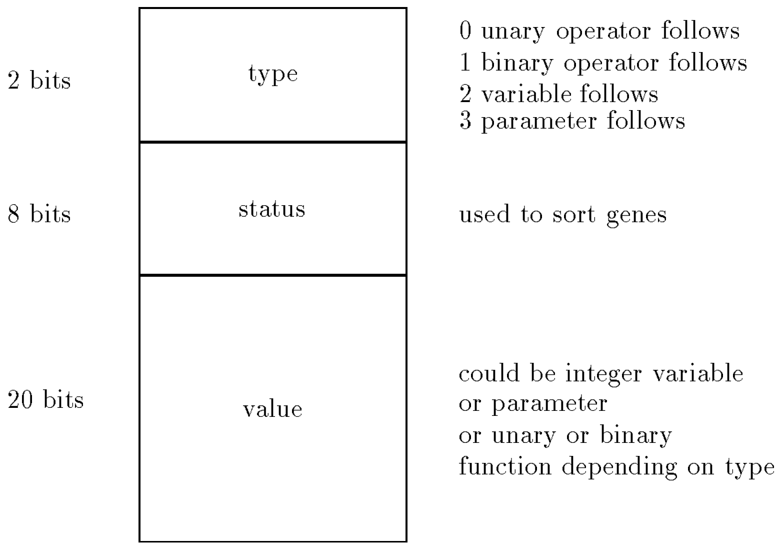
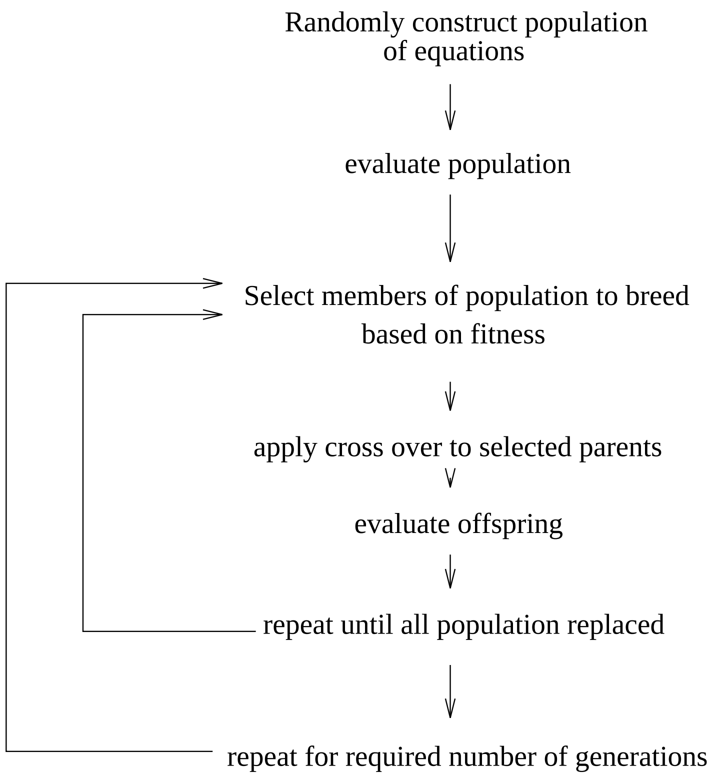


Figure 2: A more complex model representation scheme

more direct. There is no need for a bit string that has to be decoded to become a model equation. Instead the basic genetic operators of crossover and mutation are applied directly to the symbolic equations that explicitly represent the model. The trick is to ensure that these symbolic equations are manipulated in such a



Randomly construct population of equations
evaluate population
Select members of population to breed based on fitness
apply cross over to selected parents
evaluate offspring
repeat until all population replaced
repeat for required number of generations

Figure 3: The genetic programming algorithm

way that valid models are always produced. This greatly simplifies the search task and should in principle yield a much more efficient model breeding machine. Koza (1992) based his GP on LISP S expressions (which he terms programs) and this is the key to understanding the tremedous flexibility provided by GP in an automated modelling context. Figure 4 illustrates this process using crossover on a simple S expression.

The genetic programming was carried out using two approaches, one a traditional method based on LISP S-expressions (Koza, 1992) and the second used a stack based representation that seemed to offer some benefits (Perkis, 1994). Both programs were written in standard FORTRAN 77 for convenience, since this allowed implementation on various high performance computer hardware.

A FORTRAN implementation may seem a little unusual but it is very

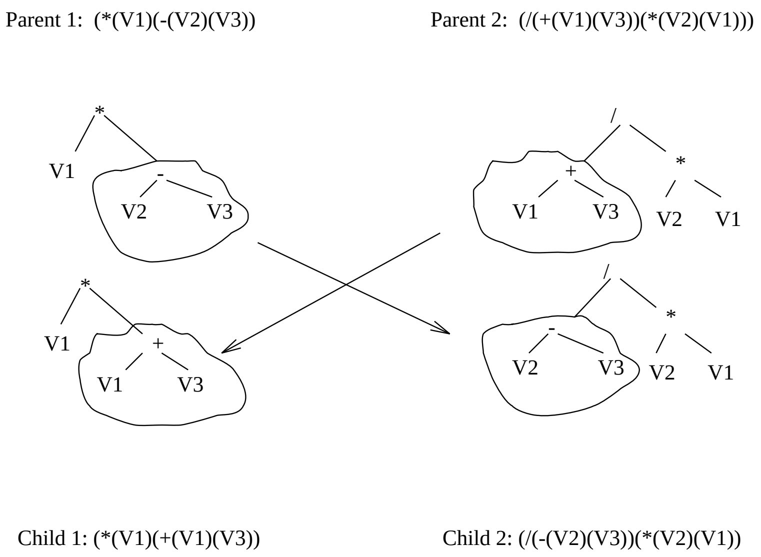

Parent 1: (*(V1)(-(V2)(V3))
Parent 2: (/(+(V1)(V3))(*(V2)(V1)))
*
V1
-
V2
V3
/
+
V1
V3
*
V2
V1
*
V1
+
V1
V3
/
-
V2
V3
*
V2
V1
Child 1: (*(V1)(+(V1)(V3))
Child 2: (/(-(V2)(V3))(*(V2)(V1))

Figure 4: Example of crossover on S expressions

straightforward. The LISP equations are handled as character strings which can only be crossed over at certain positions which generate well formed substrings, thereby completely emulating the LISP tree structure syntax. The equations contained in these character strings are then compiled into an efficient form for ease and efficiency of implementation. In this case the model is decomposed into a serial set of vector operations designed to maximise computational performance on large databases.

Figure 5 illustrates this process. An S expression in prefix form can be viewed as an infix expression. For ease of computation it is compiled into a quadruple structure. Note that here each of the operators work on vectors of data items; for example, t1=v2+v3 is actually implemented as t1(i)=v2(i)+v3(i) i=1.... N; where N is the number of spatial zones or points. The next step is to tidy up this code to remove redundant expressions, to detect constants, and to apply standard arithmetic optimisation procedures (see Bergmann, 1994). This is important because the success of this GP approach, crudely put, depends on how million models can be evaluated per hour! The GP can be parallelised in various ways. The simplest is to exploit the vectorisation in the quadruple structure.

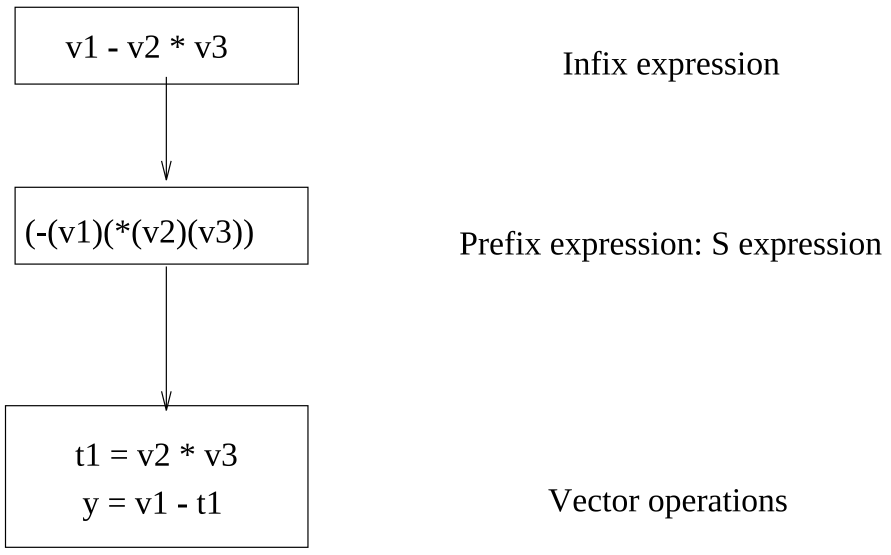


Figure 5: Infix equation, S expression and psuedo vector code

3.2 Parallel GP

The code was initially run on a Cray Y-MP and a Fujitsu VPX1200 vector supercomputer but whilst good levels of vector performance were obtained, it was quite clear that far more compute power was needed. It was subsequently ported onto the Cray T3D 512 node parallel supercomputer at Edinburgh University. The GP algorithm is naturally parallel because each member of the population of equations can be evaluated concurrently, see Figure 3. This requires that the population size is some integer multiple of the number of available processors. The initial code was parallelised in a data parallel form using CRAFT. The serial GP code could be parallelised at the vector loop level (but there was not much work here for a powerful highly parallel machine) or at the string evaluation (model level). The latter is best since there is considerable computation going on here with a non-linear optimiser being run to estimate values for the unknown parameters. Unfortunately, the compute times for each model equation are highly variable as it is a function of model complexity, the number of parameters to be estimated, and the nature of the mathematical function pieces used (e.g. a log takes much longer to compute than a multiply). This unevenness results in a large amount of idle processor time due to poor load balancing. It was obvious that a different form of parallel GP was needed if further progress was to be made.

It is necessary to develop a version of GP that uses what might be termed an asynchronous GP rather than the standard synchronous one; see Figure 6. This would allow a Message Passing (MPI) version to be developed that would ensure high levels of load balancing. The principal changes made to suit MPI are twofold: (1) the need to have a population size greater than the available number of processors being used (this is not a problem given the current trend towards highly (rather than massively) parallel systems) and the need for larger population sizes to ensure GP efficiency; and (2) the population updating needs to occurs asynchronously whereas in the serial GP it would be done synchronously when the complete population of equations had all been evaluated and their fitness ascertained. With this asynchronous approach almost perfect load balancing is achieved since as soon as a processor has finished evaluating an equation it is given another to work on using the latest fitness information available at the time. The task farm form of message passing parallel programming is very efficient at dealing with uneven computational tasks provided the individual tasks take more time than the communication overheads. In other words, the parallelism has to be relatively coarsely grained, which it certainly is in this modelling application.

3.3 Modified parallel GP

Other changes to the standard Koza (1992) form of GP are also necessary. A major departure was the replacement of the ephemeral constant by a parameter, the value of which is optimised using an embedded non-linear parameter estimation procedure. This increased execution times by a factor of between 100

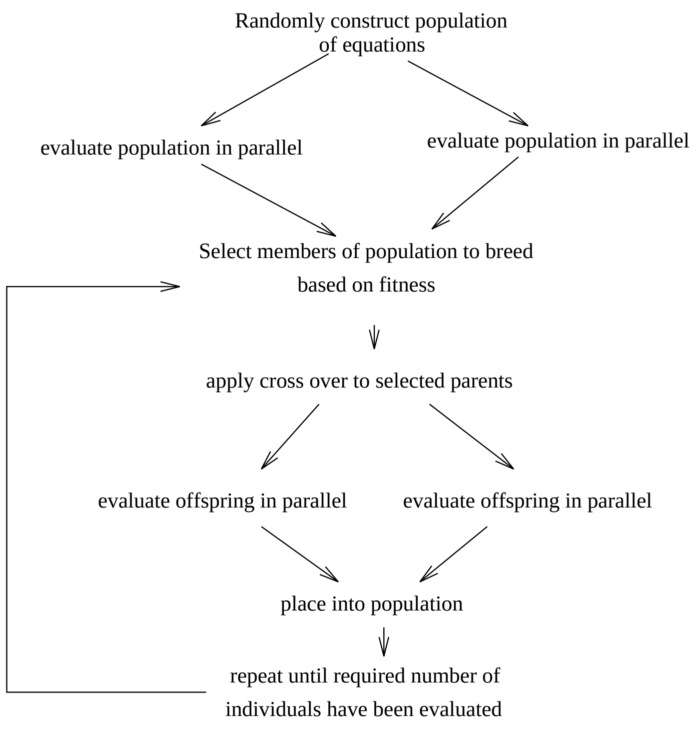

Randomly construct population of equations
evaluate population in parallel
evaluate population in parallel
Select members of population to breed based on fitness
apply cross over to selected parents
evaluate offspring in parallel
evaluate offspring in parallel
place into population
repeat until required number of individuals have been evaluated

Figure 6: The parallel genetic programming algorithm

and 1000 times but it allowed the GP to concentrate on finding a good equation instead of also having to find optimal parameter values. It seemed quite unreasonable to expect the GP to do everything! This is also useful because it avoids a potentially good model being rejected because it has poor parameter values. However, the need to use a non-linear optimiser causes a number of additional difficulties in particular: (1) a risk of finding suboptimal solutions because of any underlying assumptions of continuously differentiable functions and parameter spaces need not apply; (2) arithmetic problems due to overflows, underflows, library exceptions and NaNs which can easily happen if you assemble a random equation with a divide by zero or a negative log function argument, the GP has to learn to avoid models with these problems implicit in them as part of the model building task rather than be presented with artificially protected versions of the functions; and (3) problems of computational efficiency since the non- linear optimiser used numerical derivatives which means that the computer code used to represent the equation has to be very efficient, since it is not unusual to perform 1000 or more equation evaluations with different parameter values, for datasets containing several thousand cases every time a new model is being evaluated. The non-linear optimisation used is based on a hybrid simulated evolution and quasi- Newton method. It is very straightforward with the simulated evolution method of Schwefel (1995) being used to provide good starting values for the quasi-Newton optimiser to fine tune. Both were hardened to handle arithmetic problems. This permits the parameter and GP optimisation process to continue without propagating erroneous results.

A final consideration is the need to optimise performance levels. Careful tuning of the code on a single processor resulted in a dramatic speed-up of about 140 times on a sample of benchmark equations. Most of the improvements came from the use of vector versions of the standard mathematical functions, use of BLAS routines wherever possible, and loop unrolling.

4 A Spatial Interaction Modelling Case Study

4.1 Data and model Pieces

The spatial interaction model is widely used to describe and predict flows of people, money, goods, migrants, etc. from a set of geographically distributed origins to a set of destinations. The modern form of these mathematical models was created by Wilson (1971) over 25 years ago and, from a broader historical perspective, their structure has changed greatly since their invention 150 years ago. The task of creating genuinely new and totally different models of spatial interaction has proven to be hard and there has been little significant progress since the 1970s.

The models used here are made up of the set of terminals as shown in Table 1. This reflects a desire to model spatial interaction data, such as journey to work flows between a set of origin and destination zones. The model pieces include those typically found in spatial interaction models so that the GP could re-discover the conventional model if relevant. The variables used are shown in

Table 1, along with the functions and operators that are also available to the GP runs. Most are self- explanatory; the competing destinations term is defined as the sum of competing destinations divided by their distance from an origin. The intervening opportunity term is expressed as a count of intervening destinations between each trip-pair. The data consists of a set of car sales for one of the Standard Metropolitan Statistical Areas of Seattle, in the United States, with 86 census tracts and 35 car dealers. The travel cost is measured as the drive time in minutes.

Table 1: Spatial interaction model pieces

Terminals		
	O	Origin size
	D	Destination size
	C	Travel cost
	X	Intervening opportunities term
	Z	Competing destinations term
Operators		
	$+, -, *, /, **$	
Functions		
	sqrt, log, exp	

4.2 GP runs

For operational convenience on the Cray T3D the parameters for the model breeding varied according to the size of the job being executed. When 128 processors were used, the populations size was set at 2000, with the number of generations set at 100. For smaller runs, such as using 64 processors, the size of the task was reduced accordingly, for example evaluating 50 generations of 2000 population members, or 100 generations of 1000 population members. Several runs of each were undertaken and the best results recorded.

4.3 Results

Table 2 illustrates that the GP bred models yield a small, though improved level of performance, again measured by the sum of squares error function, which is encouraging considering the experimental nature of the exercise. The models are also quite varied in their form, which is a reflection of not only differences between the number of potential model pieces, but also that the spatial interaction data exhibits more complex, non-linear relationships.

As a consequence, the interpretability of the GP results is problematic, with complex models being generated that are not readily understood. The models do not provide quick indicators of the stronger relationships within the data, and need a significant amount of simplification before they could be written in

Table 2: Spatial interaction model breeding results

Conventional models	*Error*
$T = O.D.C^{\beta}$	9.13
$T = O.D.\exp(\beta.C)$	9.21
$T = O.D.\exp(\beta.\sqrt{C})$	9.10
$T = O.D.\exp(\beta.\log(C))$	9.14
$T = O.D.\exp(\alpha.X + \beta.C)$	9.17
$T = O.D.Z^{\delta}\exp(\beta.C)$	9.08
GP Models	*Error*
$T = \left[\left(\frac{O}{6.01Z^{-1.21}D^{-3.97}}\right) + \left(\frac{D}{-7.52O-9.36}\right)\right].71,44C^{-0.77}$	8.06
$T = 11.58\left(\frac{D.C}{Z}\right)^{-1.13}.\left[13.41D + \frac{O.D^{1.14}}{C^{0.59}} + \frac{O-D}{C^{0.81}}\right] + O.C^{0.63} + 0.38$	7.93
$T = Z^{-0.8}\left[\left(\frac{D}{(O+2C)^{4.34}} + O - \log D\right).C^{1.05}\right] - \left[C^{-0.72}.\left(\frac{O.D}{C^{0.04}} + \frac{X-D}{1.22}\right)\right]$	8.05

the form they possess in Table 2. These results suggest that a more rigorous investigation of the potential of breeding spatial interaction models over a varied number and type of data sets is necessary if the full potential of the method is to be realised.

5 Subglacial Water System Case Study

5.1 Data and Model Pieces

Our understanding of the basal hydraulic system of glaciers and ice masses has traditionally been developed from theoretical models based on our physical understanding of the basal system (e.g., film flow, channelised flow, linked cavities, canals) with rather few observations allowing direct verification of model validity. With the advent of hot-water drilling technology it has become possible to densely instrument the glacier bed; as a result while the data sets that are available are still comparatively small it is clear that we are rapidly moving from an age of data sparsity to data richness.

During the summer of 1992 the bed of Trapridge Glacier, Yukon Territory was densely instrumented with sensors to measure hydrological parameters including pressure transducers which included five installed approximately transverse to glacier flow and approximately 5m apart. Two of these sensors were installed in bore-holes that connected to the basal water system of the glacier, and as a result these sensors were dubbed C1 and C2 (Murray and Clarke, 1995). Two other sensors were installed into bore-holes that did not drain on reaching the glacier bed, and were therefore assumed to reach unconnected regions of the glacier bed; these sensors were dubbed U1 and U2 respectively. The fifth sensor appeared to be installed into a region of the bed that alternated between being in the connected and unconnected regions and so became dubbed A1. The pressures measured by these sensors have been described in detail by Murray and Clarke (1995). Important features of the data are (1) both the connected sensors (C1 and C2) and unconnected sensors (U1 and U2) show a diurnal response that we assume is forced by surface melt variations; (2) the connected sensors are out-of-phase with the unconnected sensors – at times when the pressure is high in the connected system it is low in the unconnected system, and vice versa; (3) the alternating sensor (A1) shows a semi-diurnal response such that when the pressure in the connected system is high the pressure measured at the alternating sensor is high, however when the pressure at the unconnected sensor is high the pressure measured by the alternating sensor is also high. These responses are clear during the summer 1992 data and continue to be displayed, at least during the early part of the winter period when the water system is presumably closing due to a lack of surface water input.

Taking the pressure recorded by sensor C1 in the connected system as the forcing Murray and Clarke (1995) derive a series of models to predict the water pressure within each of the three systems based on low-order differential equations. Using the final forms of these models they then describe some of the processes that they feel are important in driving the form of these models.

The models used here are made up from the set of terminals shown in table 3. The pressure record from sensor C1 is the only input variable for each model for compariability to the black box models of Murray and Clarke (1995).

Table 3: Subglacial system model peices

Terminals		
	C1	Pressure record
Operators		
	$+, -, *, /$	
Functions		
	sqrt,log,exp,sin,cos	

5.2 GP runs

For operational convenience on the Cray T3D the parameters for the model breeding varied according to the size of the job being executed. When 128 processors were used, the populations size was set at 1000, with the number of generations set at 100. For smaller runs, such as using 64 processors, the size of the task was reduced accordingly, for example evaluating 50 generations of 1000 population members, or 100 generations of 500 population members. Several runs of each were undertaken and the best results recorded.

5.3 Results

Table 4 shows the results obtained by GP compared to the blackbox models derived by Murray and Clarke (1995).

Table 4: Subglacial Modeling Results

Connected System		
Black Box	$R(t) = 1.01F(t) - 4.96$	0.2m
GP	$R(t) = F(t) - 3.75$	0.3m
Unconnected System		
Black Box	$a_2 \frac{d^2R}{dt^2} + \frac{dR}{dt} - a_0 [R^\star - R(t)]$ $= b_1 \exp[D_0 F(t)] \frac{dF}{dt}$	2.1m
GP	$R(t) = \frac{0.86}{e^{0.04f(t)}} - 0.05.(-1384.5 - F(t))$	3.4m
Alternating System		
Black Box	$R(t) = F(t_i^s) + c_1 [F(t) - F(t_i^s)]$	
or	$a_2 \frac{d^2R}{dt^2} + \frac{dR}{dt} - a_0 [R^\star - R(t)] =$ $\{b_1 \exp[D_0 F(t)] \frac{dF}{dt} + b_0 [F(t) - R(t)]\}$	6.1m
GP	$R(t) = 3F(t) + \frac{18000}{F(t)} - \frac{242423}{F^2(t)} - 374$	3.1m

Again the GP results produce a better preformance and in this case while complex can be considered no worse than the black box models. It should also be noted that the black box model for the alternating system contains hardwired switches between the two states where as the GP model has no knowledge added.

6 Conclusions

From a GP perspective there is a feeling that the best achievable results have still not being produced. Whether this reflects a lack of sufficient compute power to investigate larger population sizes or a need for a more efficient parameter estimation procedure, problems with the GP itself are a matter for further investigation and debate. Indeed debugging a GP is actually quite hard because the GP will cleverly accommodate all manner of non-fatal errors in the code. Indeed a major research effort is probably needed to determine good ways of validating GP software. Nevertheless, the paper clearly provides a glimpse of the potential of GP as a generic tool for creating new models of complex geographical systems. In principle it would appear to offer the ultimate technology for implementing an inductive approach to knowledge creation from data. At a time when there is a vast explosion in all kinds of information it is useful to know that GP provides the basis for new sets of tools that are able, in principle, to convert at least some of these data riches into new knowledge. Whether this approach works well depends on the speed of the available high performance computers, the power of the GP method itself, and the subsequent skills of the modeller in interpreting the equations that are created and declared optimal.

References

Barrow, D. (1993) The use and application of genetic algorithms, *Journal of Targeting, Measurement and Analyses for Marketing* 2: 30-41.
Goldberg, D E. (1989) *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison Wesley, Reading Mass.
Holland, J. (1975) *Adaptation in Natural and Artificial Systems*, the University of Michigan Press, Ann Arbor.
Koza, J. (1992) *Genetic Programming: on the programming of computers by means of natural selection*, MIT Press, Cambridge, Mass.
Koza, J. (1994) *Genetic Programming II: Automatic Discovery of Reusable programs*, MIT Press, Cambridge, Mass.
Murray, T. and Clarke, G.K.C. 1995. Black-box modeling of the subglacial water system. *Journal of Geophysical Research*, 100(B7), 10231–10245.
Openshaw, S. (1976) An empirical study of some spatial interaction models, *Environment and Planning A* 8: 23-41.
Openshaw, S (1986) Modelling relevancy, *Environment and Planning A* 18: 143-147
Openshaw, S. (1988) Building an automated modelling system to explore a universe of spatial interaction models, *Geographical Analysis* 20: 31-46.
Openshaw, S. (1992) Some suggestions concerning the development of artificial intelligence tools for spatial modelling and analysis in GIS, *Annals of Regional Science* 20: 35-51
Openshaw, S. (1996) Fuzzy Logic as a New Scientific Paradigm for doing Geography, *Environment and Planning A* (forthcoming).
Perkis, T. (1994) Stack-based genetic programming, *IEEE World Congress on Computational Intelligence*
Schwefel, H.P. (1995) *Evolution and optimum seeking*, Wiley, New York
Wilson, A.G. (1971) A family of spatial interaction models and associated developments, *Environment and Planning* A 3: 1-32