Building New Spatial Interaction Models Using Genetic Programming

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

Abstract

Building computer models of human geographic systems is difficult because of the poverty of applicable theory that can be used to specify well formed and soundly based mathematical models. As a result there has been little progress in this area since the late 1960's when mathematical model building procedures based on entropy-maximising methods were first applied to spatial interaction data. In the mid–1980's, attention was focused on the use of genetic algorithms to explore the universe of different models that could be built from the available symbolic pieces; that is there is a very very large number of permutations of the model building blocks; viz. a mix of unary and binary operators, unknown parameters and observed variables, and rules of syntax for well formed equations. The so-called Automated Modeling System (AMS) worked but never fulfilled its expectations. The paper revisits the problem but re-casts the AMS approach into a genetic programming framework. Some preliminary results based on Cray-YMP runs are reported.

1 Introduction

This paper develops the work of Openshaw (1988) who described the exploration of the "universe" of possible spatial interaction models by the use of the now classic genetic algorithms (GA) as developed by Holland (1975).

The principal problem is that of how best to represent symbolic equations incorporating the available range of model pieces by a fixed length bit string that would allow the GA maximum freedom to search the universe of possible models.

Two different representational schemes have previously been 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 that these were estimated by a nonlinear least squares procedure.

A more complex alternative which was used in an early version of OMIGA (Barrow 1993) that was based on a subsequent development of (Openshaw 1988) (1988). Figure 2 outlines this model representation. A model consists of a number of these genes. The genes are input into a reverse polish stack sorted by their status value. The model equation that emerges is the longest complete equation that satisfies reverse polish logic. The problem is that this type of implementation is hard for the GA to handle, due to the possibility of redundancy, its variable length and its self defining nature. However, it does work, although there is a feeling that better results might be obtainable.

In this work we have attempted to improve on AMS by using the newer technique of genetic programming (GP) (Koza 1992). This method of representation is much more direct. There is no need for a bit string representation. The symbolic equations are manipulated in such a way that valid models are always produced.

2 Genetic Programming

The basic algorithm of genetic programming is shown in figure 3.

The genetic programming was carried out using two approaches, one a traditional method based on LISP S–expressions (Koza 1992) and the second based on a stack based representation (Perkis 1994). Both programs were written in standard FORTRAN77 since this provided most support for parallel implementation on vector parallel machines.

A major change compared with Koza (1992) was the replacement of the ephemeral constant by a parameter, the value of which is optimised by an embedded nonlinear parameter estimation procedure. This increased execution times by a factor of between 100 and 1000 times but it reduced the load on the GP to not only find a good equation but also optimal parameter

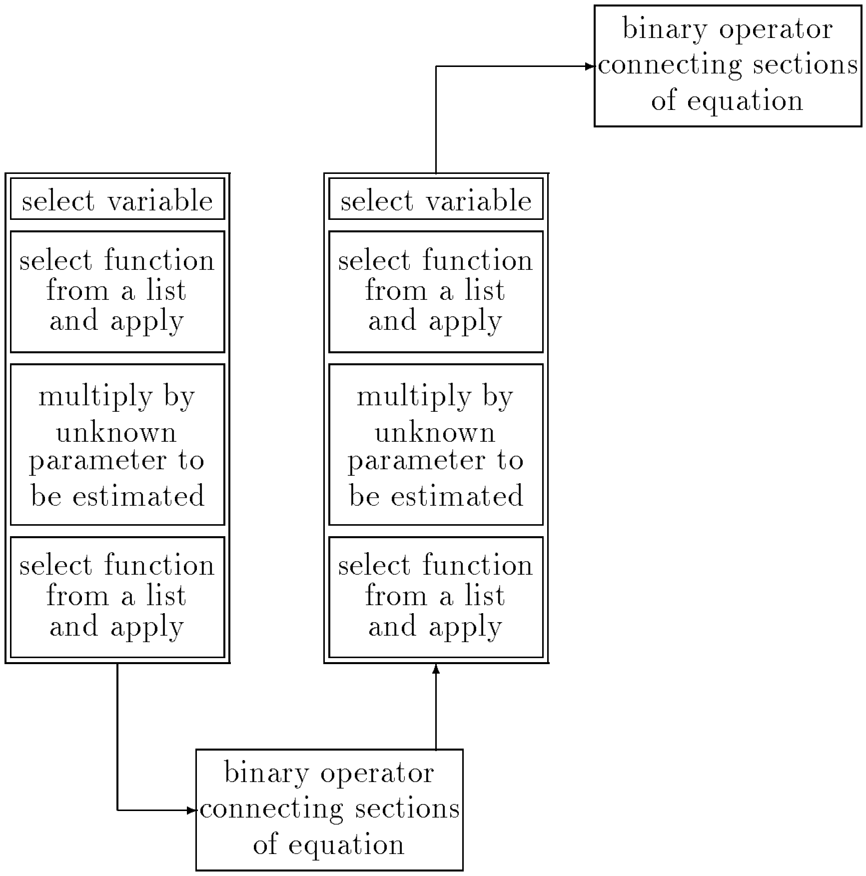

Figure 1: A simple model representation scheme

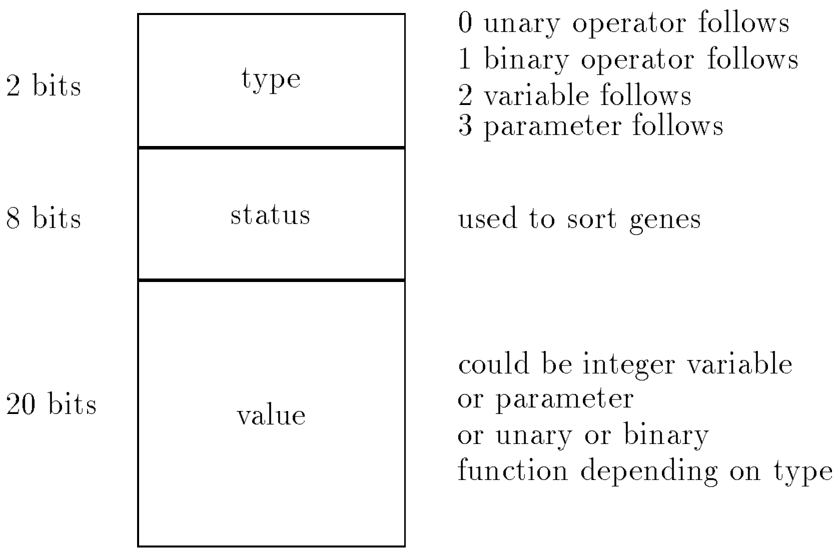

Figure 2: A more complex model representation scheme

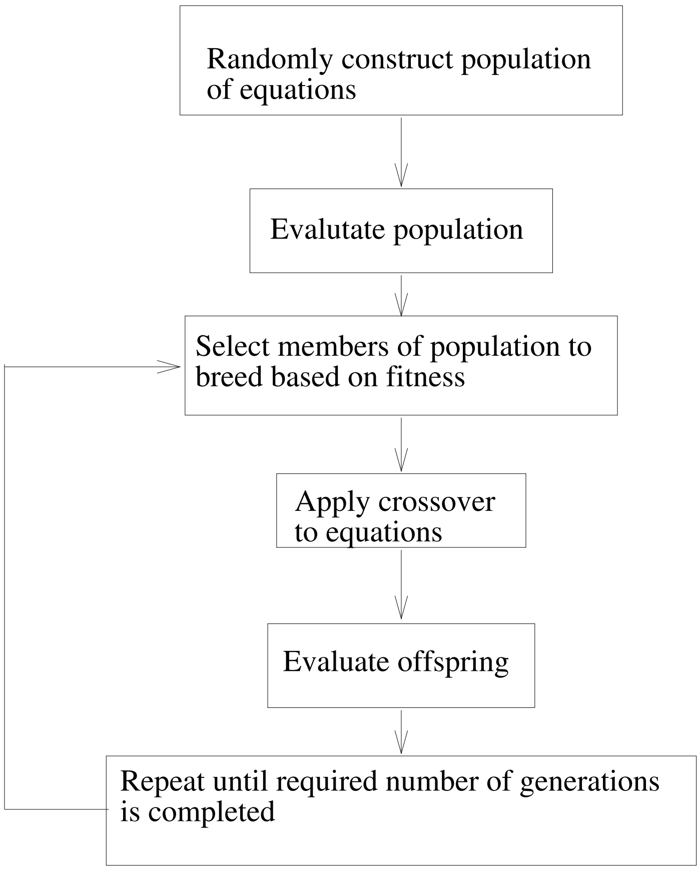

Figure 3: The genetic programming algorithm

Table 1: Function Set Used

$+$	addition
$-$	subtraction
$*$	multiplication
$/$	division, protected to prevent divide by zero
log	natural logarithm, protected against zero input
exp	exponential function
minus	unary minus
csum	column sum for the matrix
rsum	row sum for the matrix
numeric constants	defined randomly at creation and then optimised

values.

The models developed by the program were made up of the set of terminals is 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 found in spatial interaction models so that the GP could "re–discover" the conventional model if relevant. The models normally used to represent these data are nonlinear and have the following form

$$T_{ij} = O_i D_j A_i B_j f(C_{ij}) \tag{1}$$

$$A_i = \frac{1}{\sum_j B_j D_j f(C_{ij})} \tag{2}$$

$$B_i = \frac{1}{\sum_i O_i A_i f(C_{ij})} \tag{3}$$

where
T_{ij} is the number of trips between i and j
O_i is the number of trips starting in zone i
D_j is the number of trips ending in zone j
A_i and B_j are optional and ensure either singly or doubly constrained predicted T_{ij}s are produced
$f(C_{ij})$ is a function applied to the cost (C_{ij}) of traveling from zone i to j

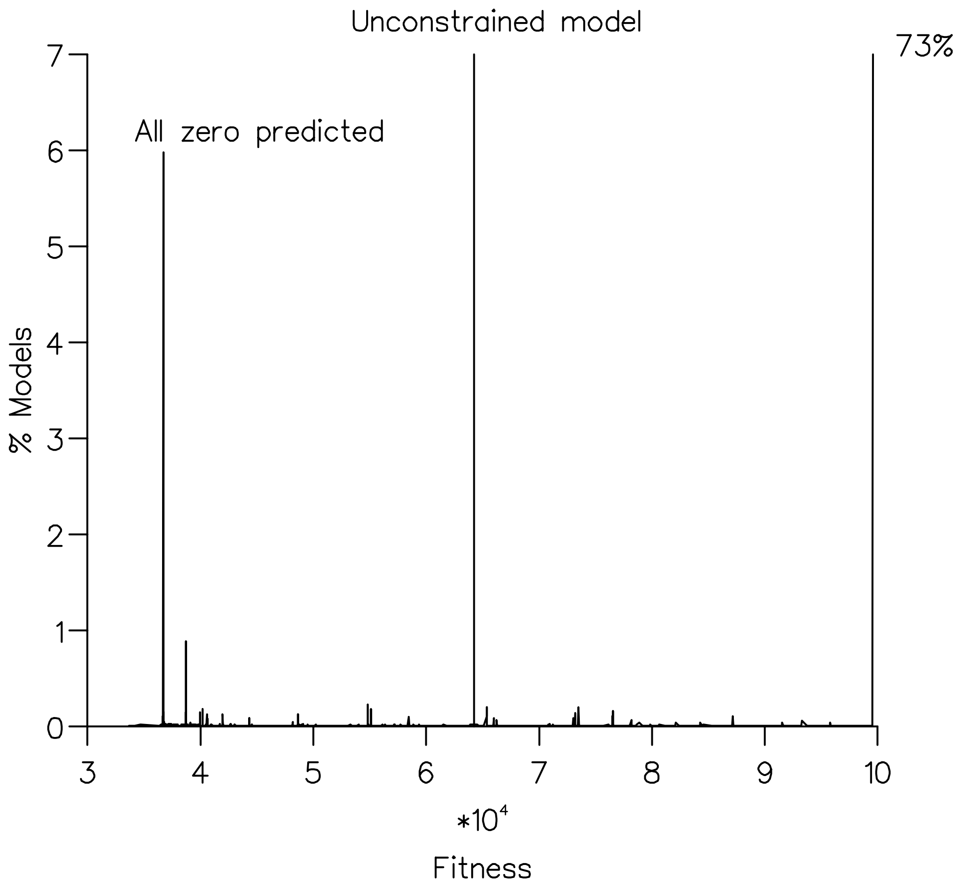

Figure 4: Fitness of 10000 random models

2.1 How fit is that model?

The initial fitness function that was chosen was a simple sum of absolute errors:

$$\mathbf{F} = \sum_{i=0,j=0}^{i=73,j=73} |\mathrm{Obs}(i,j) - \mathrm{Pred}(i,j)| \tag{4}$$

However it soon became clear that this was not a useful fitness function since the GP discovered that a model could score a reasonably good score simply by predicting zero for all trips (see figure 4). This had not previously been commented upon and was an interesting albeit not particularly useful discovery. Therefore a new fitness function was developed.

$$\mathbf{F} = \sum_{i,j=0}^{i,j=N} \left| \text{Obs}(i,j) - \left(\text{Pred}(i,j) \times \frac{\sum \text{Obs}}{\sum \text{Pred}} \right) \right| \times \begin{Bmatrix} \left(\frac{\sum \text{Obs}}{\sum \text{Pred}}\right) & \text{if } \frac{\sum \text{Obs}}{\sum \text{Pred}} > 1 \\ 1 & \text{if } \frac{\sum \text{Obs}}{\sum \text{Pred}} < 1 \end{Bmatrix} \tag{5}$$

3 Results

The 73 zone dataset of Openshaw (Openshaw 1976) was used to develop and test the models. All the models developed by the program were unconstrained models and their success can be judged against the classical unconstrained model

$$T_{ij} = O_i D_j e^{-bC_{ij}} \tag{6}$$

Table 2 also shows the results of constrained models for comparison.

Table 2: Results of various models applied to the 73 zone dataset (Openshaw 1976)

	Unconstrained Model	Singly Constrained Model	Doubly Constrained Model	Trad GP	Stack GP
72 zone	64231	27988	23030	27457	24929

It is very interesting that the GP models outperform the conventional unconstrained model and come close to competing with the singly and doubly constrained alternatives.

For the 73 zone data the two best performing equations were for the traditional GP:

```
(/(V 2)(*(V 5)(log(log(V 2)))))
```

which simplifies to

$$\frac{D_j}{\text{Diag} \times \log(\log(D_j))}$$

where Diag is a flag which equals 1 when on the matrix diagonal and 0 otherwise.

This suggests that distance decay is far less important than the size of destination zone D_j.

For a stack based GP the best equation was:

```
(/)(*)(/)(V 5)(-1.80543)(-)(9.09109)(V 2)(V 5)
```

which simplifies to

$$\frac{(9.09 - D_j) \times \left(\frac{\text{Diag}}{-1.81}\right)}{\text{Diag}}$$

where Diag is defined as above, note also that divide by zero is protected to give zero.

This model also emphasises the D_j term. This is surprising since all the models of journey to work flows have always focused their attention on distance decay effects. For example in equation 6 the $e^{-bC_{ij}}$ factor. The importance of D_j only model is that it suggests the pattern of destination opportunities is actually more important than distance. This is certainly worthy of further investigation. Whether these models have any theoretical significance must depend on whether the same structure of model is robust and performs well on several datasets. That is a task for further study.

4 Conclusion

From a GP perspective there is a feeling that optimal results are still not being produced. Whether this reflects lack of sufficient compute power to investigate larger population sizes or problems with the GP used is 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. Indeed a major research effort is probably needed to determine good ways of validating GP software.

References

Barrow, D. (1993). The use and application of genetic algorithms, *Journal of Targeting, Measurement and Analysis for Marketing* **2**: 30–41.

Holland, J. (1975). *Adaption in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor.

Koza, J. (1992). *Genetic Programing: on the programming of computers by means of natural selection*, MIT Press, Cambridge, Mass.

Openshaw, S. (1976). An empirical study of some spatial interaction models, *Environment and Planning A* **8**: 23–41.

Openshaw, S. (1988). Building an automated modeling system to explore a universe of spatial interaction models, *Geographical Analysis*.

Perkis, T. (1994). Stack–based genetic programming, *IEEE World Congress on Computational Inteligence*.