Genetic Algorithms for Simultaneous Equation Models

José-Juan López-Espín
Universidad Miguel Hernández (Elche, Spain)
Domingo Giménez
Universidad de Murcia (Murcia, Spain)

Contents

- Introduction
- Simultaneous equations models
- OLS and 2SLS techniques
- The problem: Find the best SEM given a set of values of variables
- Genetic Algorithms for selecting the best SEM
  - Defining a valid chromosome
  - Initialization and EndConditions
  - Evaluating a chromosome
  - Crossover
  - Mutation
- Greedy Method
- Experiment results
- Conclusions and future works

Introduction

- S.E.M. has been used in econometrics for years.
- Traditionally, Simultaneous Equation Models (SEM) have been developed by people with a wealth of experience in the particular problem represented by the model.
- The objective is to develop an algorithm which, given the endogenous and exogenous variables, finds the best SEM possible
- The space of the possible solutions is very large and exhaustive search methods are not well suited here.
- A combination between genetic and greedy methods is studied.

Simultaneous Equations Models

The scheme of a system with *N* equations, *N* endogenous variables, *K* exogenous variables and *d* sample size is (*structural* form)

$$Y_1 = \beta_{12}Y_2 + \beta_{13}Y_3 + \ldots + \beta_{1N}Y_N + \gamma_{11}X_1 + \ldots + \gamma_{1K}X_K + u_1$$
$$Y_2 = \beta_{21}Y_1 + \beta_{23}Y_3 + \ldots + \beta_{2N}Y_N + \gamma_{21}X_1 + \ldots + \gamma_{2K}X_K + u_2$$
$$\ldots$$
$$Y_N = \beta_{N1}Y_1 + \beta_{N2}Y_2 + \ldots + \beta_{NN-1}Y_{N-1} + \gamma_{N1}X_1 + \ldots + \gamma_{NK}X_K + u_N$$

where Y_i, X_j and u_i are *dx1* $i = 1 \ldots N, j = 1 \ldots K$

These equations can be represented in matrix form

$$BY + \Gamma X + u = 0$$

OLS (Method)

OLS (**Ordinary Least Square**) can be used to solve a regression model

$$Y_t = \alpha_1 X_{1t} + \ldots + \alpha_n X_{nt} + u_t \qquad t = 1 \ldots d$$

In matrix form

$$Y = \beta X + u$$

The expression of the estimator is

$$\hat{\beta} = (X^t X)^{-1} X^t Y$$

2SLS (Two Step Least Squares)

PROBLEM

- OLS can not be used in structural form because random variable and endogenous variables are correlated.

$$Y_i = \beta_{i1} Y_1 + \ldots + \beta_{iN} Y_N + \gamma_{i1} X_1 + \ldots + \gamma_{iK} X_K + u_i$$

SOLUTION

- Endogenous variables are replaced for approximations (proxys variables).
- The proxy of Y is calculated using OLS with Y and the exogenous in the system.

$$\hat{Y} = \beta X + v$$

- When the endogenous have been replaced, OLS is used again to solve the equations.

$$Y_i = \beta_{i1} \hat{Y}_1 + \ldots + \beta_{iN} \hat{Y}_N + \gamma_{i1} X_1 + \ldots + \gamma_{iK} X_K + u_i$$

The problem: Find the best SEM given a set of values of variables

- One model is considered better than another if its criteria parameter is lower than that of the other.
- AIC is one of the most used methods for comparing models.

$$AIC = d \ln |\hat{\Sigma}_e| + 2 \sum_{i=1}^{N} (n_i + k_i - 1) + N(N+1)$$

- *d* is sample size, n_i and k_i the number of endogenous and exogenous variables in the equation *I, and* $\hat{\Sigma}_e$ *is* the covariance matrix of the errors

Genetic Algorithms for selecting the best SEM

- Each chromosome represents a candidate to be the best model.
- A chromosome is defined as a matrix with *N* rows and *N+K* columns.
- In each row, an equation is represented using ones and zeros.
- If variable *j* appears in equation *i*, the value for the *(i,j)* position in the chromosome is one, and zero if not.
- The *N* first columns of a chromosome represent the endogenous variables and the other *K* columns represent the exogenous ones.

For example, in a problem with *N=2* endogenous variables (Y_1 and Y_2) and *K=3* predetermined variables (X_1, X_2 and X_3):

$$y_1 = \gamma_{1,1} x_1 + \gamma_{1,2} x_2 + \beta_{1,2} y_2 + u_1$$

$$y_2 = \gamma_{2,3} x_3 + \beta_{2,1} y_1 + u_2$$

$\longrightarrow$

11001

11110

Scheme of a genetic algorithm

Initialize(S)
WHILE Not End Conditions(S)
    Evaluate(S)
    SS1=Select the best ranking of S
    SS2=Crossover and Mutation(SS1)
    S=IncludeSolutions(SS2)

END WHILE

Defining a valid chromosome

- The model has to have at least one equation.
- If the *(i,i)* element is zero, the column *i* will have only zeros.
- Each equation in the model must have at least two variables.

- Rank condition: Equation *i* is identified if it is possible to find a *(N-1) x(N-1)* matrix with full range where the columns are the unknown variables

$$\gamma_{1,1},\ldots,\gamma_{N,K},\beta_{1,2},\ldots,\beta_{N,N-1}$$

that do not appear in the equation.

Initialization and EndConditions

- Each chromosome is generated according to algorithm in the right.
- The population size (called *PopSize*) is stated at the beginning.
- The process is repeated until the process reaches the maximum of iterations called *MaxIter* or the best fitness is repeated over a number of successive iterations, called *MaxBest*.
- Both parameters are stated at the beginning of the algorithm.

1. **GENERATE** the $N(N+K)$ elements randomly (zeros and ones with the same probability)

{C1 AND C2 CONDITIONS}

2. **IF** N or $N-1$ elements $e_{(i,i)}$ are zero with i=1,...,N
3. invert all the elements $e_{(i,i)}$ with i=1,...,N
4. **END IF**

{C3 CONDITION}

5. **FOR** i=1...N
6. **IF** the element $e_{(i,i)}$ is zero
7. make zero all the elements in the column i
8. **END IF**
9. **END FOR**

{C4 CONDITION}

10. **FOR** i=1...N
11. **IF** the equation i fails the range condition
12. generate randomly this equation (row i) and go to 2.
13. **END IF**
14. **END FOR**

Evaluating a chromosome

- Algorithm in the right shows the scheme of the fitness function of a chromosome.
- Matrices Y_c and X_c are formed by the columns of Y and X corresponding to the variables in chromosome *c*.

1. build the system using the chromosome c and the set of variables Y and X
2. **COMPUTE** $\hat{Y}_c = X_c(X_c^t X_c)^{-1} X_c^t Y_c$
3. **FOR** i=1...N_c
   {solve each equation in the system}
4. **COMPUTE** $(X_{c,eq-i}^t X_{c,eq-i})^{-1} X_{c,eq-i}^t y_{c,i}$
5. **END FOR**
6. **FOR** i=1...N_c
   {obtaining the error variables}
7. **COMPUTE** $e_{c,i}$=$y_{c,i}$- estimation of $y_{c,i}$
8. **END FOR**
9. **COMPUTE** AIC

- The cost of evaluating a chromosome is :

$$\frac{2}{3}K^3 + 2K^2(d+N) + 4NKd + 2N^2d + 3N + T_{\log} +$$

$$\sum_{i=1}^{N} \left(\frac{2}{3}(n_i + k_i - 1)^3 + 2(n_i + k_i - 1)^2(d+1) + 4(n_i + k_i - 1)d\right)$$

Crossover

- Three sorts of crossover are studied:
  - Crossover between elements
  - Crossover between equations
  - Crossover inside an equation

parents		**between elements** e = 10		**between equations** e = 1		**inside an equation** e = 2, v1 = 2, v2 = 3	
father	mother	son	daughter	son	daughter	son	daughter
11110110	10100100	11110110	10100110	11110110	10100100	11110110	10100100
11110101	1110100	11110100	1110101	1110100	11110101	11110100	1110101
1110110	11110110	11110110	1110110	11110110	1110110	1110110	11110110

problem size			crossover **between elements**			crossover **between equations**			crossover **inside an equation**		
N	***K***	***d***	t	iter	*best fitness*	t	iter	*best fitness*	t	iter	*best fitness*
10	15	50	3,03	48	2683,13	5,11	97	2732,90	0,66	20	2833,41
15	20	50	8,00	62	4548,68	6,73	53	4540,93	1,94	40	4709,50
30	40	100	58,33	50	21937,02	87,54	72	22120,10	9,47	17	22765,68
40	50	100	325,87	111	30956,78	294,19	102	31262,20	64,41	24	32975,04

Mutation

- A small probability of mutation is considered in all the iterations.
- A chromosome of the new subset generated in the crossover is chosen randomly, and a equation and a variable are generated randomly. Then, the element is inverted.

Greedy Method

- PROBLEM: When a chromosome is mutated and then situated in a different part of the set of solutions, this chromosome normally does not have

  enough quality to survive long enough to create new chromosomes in this area, and perhaps best solution is next to it.

- To avoid this problem, a greedy method is used in the mutation following the algorithm in the right.

- A chromosome is good enough when its evaluation is lower than a parameter called *SV*.

---

1. Generate *e* between 1 and *N* ramdonly.
2. *EndConditions*=**FALSE**
3. **WHILE** *Not EndConditions*
4. Generate *v* between 1 and *N+K* randomly
5. *c1*=Mutate(*c*) {invert the element *(e,v)* of the chromosome *c*}
6. **IF** GoodChromosome(c_1) **AND** Evaluation(c_1)<Evaluation(*c*)
7. **COMPUTE** *c=c1*
8. **END IF**
9. **IF** Evaluation(*c*)<*SV*
10. EndConditions=TRUE
11. **END IF**
12. **END WHILE**

Experimental Results

- Experimental results have been obtained in a Intel Itanium 2 systems connected by Gigabit Ethernet, where each node is equipped with four sets of dual-core 1.4 GHz Montecito processor, i.e. 8 processors for node.
- A comparison of the solution found by the genetic algorithm when varying the population size (*PopSize*), *N* and *K*. The sample size *d*=10, the crossover is "inside an equation.
- Execution time (in seconds) and speed-up of the algorithm in shared memory.

size		best fitness		
N	**K**	***PopSize*=100**	***PopSize*=500**	backtracking
2	2	66,44	66,44	66,44
2	3	46,18	46,18	46,18
3	3	-177,03	-214,91	-216,68
3	4	-124,05	-213,16	-216,68
4	4	-99,73	-161,67	-218,58

PopSize	**N**	**K**	**d**	**1th**	**2th**	**sp**	**4th**	**sp**	**8th**	**sp**
100	10	15	50	4,22	2,51	1,68	1,62	2,60	1,04	4,06
100	30	40	100	40,74	26,21	1,55	16,24	2,51	12,31	3,31
100	50	65	150	217,79	152,19	1,43	102,27	2,13	63,81	3,41
100	70	90	200	709,05	417,62	1,70	277,15	2,56	185,88	3,81
500	10	15	50	21,31	11,55	1,85	7,50	2,84	4,70	4,53
500	30	40	100	201,29	115,71	1,74	62,30	3,23	47,47	4,24
500	50	65	150	1065,77	699,20	1,52	368,11	2,90	229,68	4,64
500	70	90	200	3580,94	1927,76	1,86	1076,45	3,33	699,21	5,12

Conclusions and Future works

Conclusions

- An algorithm to obtain a satisfactory Simultaneous Equation Model from a set of variables is studied.
- Genetic and greedy methods are combined to avoid to fall in local minimum and to speed-up the convergence.
- A shared memory version, which allows us to efficiently use multicore processors in the solution of the problem, has been developed.

Future Works

- Application to real problems.
- Develop an hybrid (message-passing plus shared memory) algorithm.