Reference

[1] J. H. Holland, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence", 2nd Edition, The MIT Press, 1992.

[2] J. R. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection", Cambridge, MA: The MIT Press, 1992.

[3] D. A. Augusto and H. J. C. Barbosa, "Symbolic Regression via Genetic Programming", *Proc.of the 6th Brazilian Symposium on Neural Networks (SBRN'00)*, 2000, pp. 173 – 178.

[4] W. Banzhaf, J.R. Koza, C. Ryan, L. Spector and C. Jacob, "Genetic programming", *IEEE Intelligent Systems and their Applications*, vol. 15, no. 3, 2000, pp. 74 – 84.

[5] J. R. Koza, "Darwinian invention and problem solving by means of genetic programming", *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC '99)*, vol. 3, 1999, pp. 604 – 609.

[6] Microsoft Visual Studio 2010, http://www.microsoft.com/visualstudio/en-us, 2012.

[7] Microsoft .NET Framework, http://www.microsoft.com/net, 2012.

[8] AForge.NET Framework, http://www.aforgenet.com/, 2012.

[9] Microsoft SQL Server Compact, http://msdn.microsoft.com/en-us/data/ff687142.aspx, 2012.

[10] D. F. Specht, "A General Regression Neural Network", *IEEE Transactions on Neural Networks*, vol. 2, no. 6, 1991, pp. 568 – 576.

[11] S. Tomioka, S. Nisiyama, and T. Enoto, "Nonlinear Least Square Regression by Adaptive Domain Method with Multiple Genetic Algorithms", *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, 2007, pp. 1 – 16.

[12] S. Crino and D.E. Brown, "Global Optimization with Multivariate Adaptive Regression Splines", *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 2, 2007, pp. 333 – 340.

[13] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers", *IBM Journal of Research and Development*, vol. 3, no. 3, 1959, pp. 210 – 229.

[14] R. Balzer, "A 15 Year Perspective on Automatic Programming", *IEEE Transactions on Software Engineering,* vol. SE-11, no. 11, 1985, pp. 1257 – 1268.

[15] N. L. Cramer, "A Representation for the Adaptive Generation of Simple Sequential Programs", *Proc. of an International Conference on Genetic Algorithms and Their Applications (ICGA85),* 1985, pp. 183-187.

[16] R. Poli, W. B. Langdon and N. F. McPhee, "A Field Guide to Genetic Programming", 2008.

[17] J. R. Koza, "Genetically Breeding Populations of Computer Programs to Solve Problems in Artificial Intelligence", *Proc. of the 2nd International IEEE Conference on Tools for Artificial Intelligence,* 1990, pp. 819 – 827.

[18] H. Tuan-Hao, R. I. McKay, D. Essam, and N. X. Hoai, "Solving Symbolic Regression Problems Using Incremental Evaluation in Genetic Programming", *IEEE Congress on Evolutionary Computation (CEC 2006),* 2006, pp. 2134 – 2141.

[19] G.Dworman, S.O.Kimbrough, and J.D.Laing, "On Automated Discovery of Models Using Genetic Programming in Game-Theoretic Contexts", *Proc. of the 28th Hawaii International Conference on System Sciences,* vol. 3, 1995, pp. 428 – 438.

[20] GPdotNET, Genetic Programming Tool, http://gpdotnet.codeplex.com/, 2012.

[21] GPLAB, A Genetic Programming Toolbox for MATLAB, http://gplab.sourceforge.net/, 2012.

[22] MathWorks MATLAB, http://www.mathworks.in/products/matlab/, 2012.

[23] GNU Octave, http://www.gnu.org/software/octave/, 2012.

[24] World Bank Data Catalog, http://data.worldbank.org/, 2012.

[25] WEKA (Waikato Environment for Knowledge Analysis), Version 3.6.4, http://www.cs.waikato.ac.nz/~ml/weka/, 2012

Appendix A

Detailed Design Diagram

A.1 Introduction

In this section it is going to present the detailed design diagram. Figure A.1 shows a detailed diagram of the GPVLab software.

A.2 A Detailed Design Diagram of GPVLab

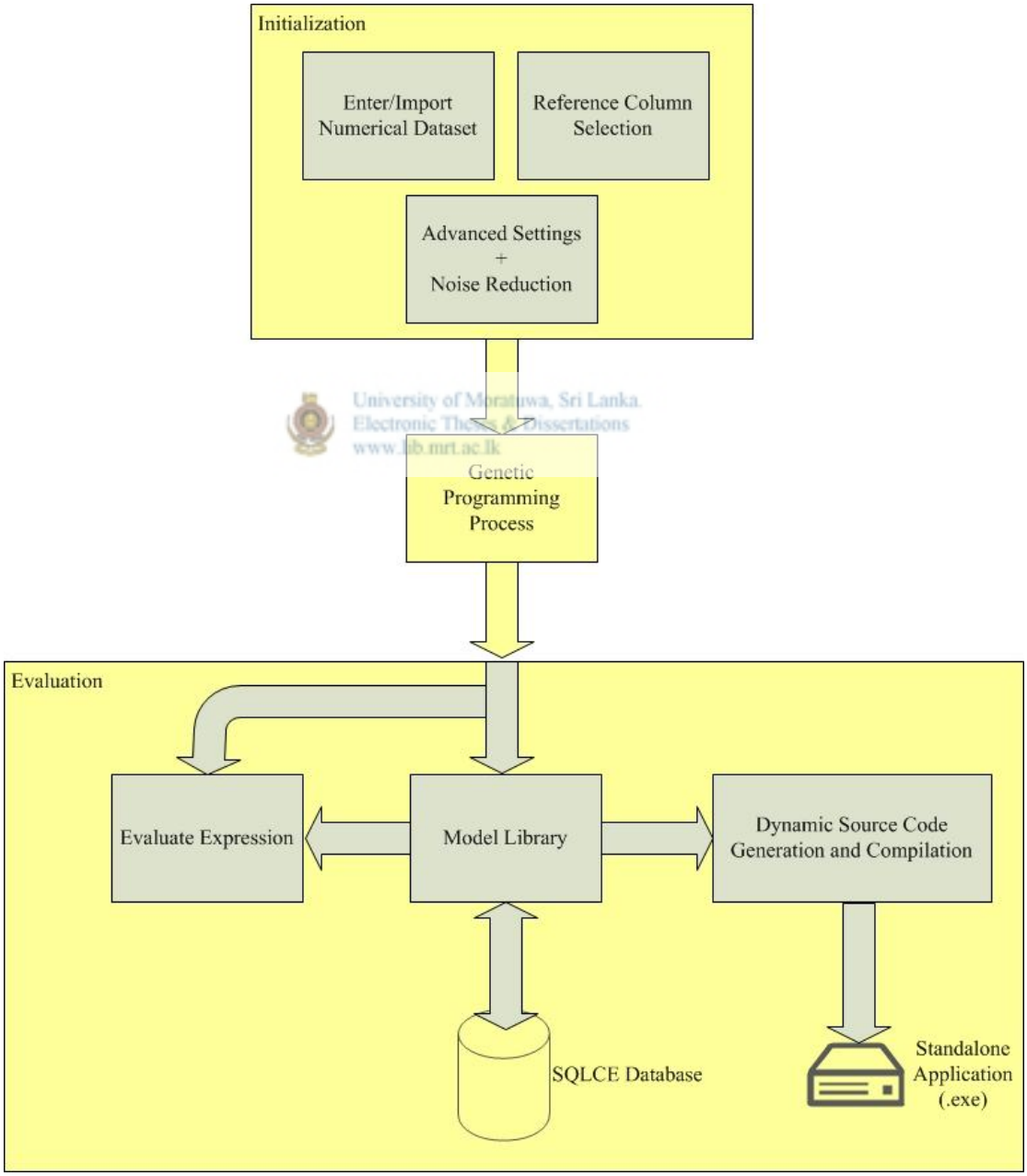

Figure A.1 – A detailed design diagram of GPVLab

Appendix B

Genetic Programming Process

B.1 Introduction

In this section it is going to describe the genetic programming process in detail. An example scenario is taken to describe the genetic programming process.

B.2 Genetic Programming Process

For an example consider the dataset shown in Table B.1. This is a hypothetical dataset that contains data about nitrogen (N), phosphorus (P), and potassium (K) content of different fertilizers which have been used on farms. The output column shows the output yield of each farm corresponding to different fertilizers.

Table B.1: Example input dataset

N	P	K	Output (Y)
2	4	1	200
4	8	2	300
6	12	3	400
8	16	4	500
10	20	5	600

Think of this as an input dataset which the system is going to use to discover a data model. According to the system it has to select the reference attribute. Think that it has selected the output column ‘Y’ as the reference column. Now remaining columns are ‘N’, ‘P’, and ‘K’ which the system is going to use to generate a model. In here ‘N’, ‘P’, and ‘K’ are the terminals. That is, in the final outcome there will be an evaluable expression which can be used to find the value of ‘Y’ by entering ‘N’, ‘P’, and ‘K’ values. So these are the inputs of the final outcome. This application tries to find the mathematical relationship between attributes. Therefore the functions of this system will be mathematical operations such as, +. -, *, /. For this example author consider only the basic arithmetic operators +,-,*, /. Normally in GP solution, it execute each and every evaluable expression in the population and evaluate how well those programs suites for the given problem. This can be done by taking the sum of absolute errors over number of fitness cases, which is the number of records in our dataset. Next the system has to determine the parameters for controlling the run.

Typically these parameters are the population size and the maximum number of generations to be run. The termination criterion of this system is the event of exceeding the maximum number of generations or finding a perfect solution. The main GP process starts with generating the initial population. Table B.2 shows an example initial population which may generate by this system.

Table B.2: Example initial population

10N+P/K
N+P*2K
2N/P+10K
4N-PK
7N*P-2K
N/5P+K

Each of these programs will execute and evaluate to find the fitness value. Usually all the programs in the initial generation have a poor fitness. Nevertheless some programs are better than others. These programs are more likely to be selected for genetic operations. Most of the better ones will survive by copying them to the next generation.

After applying cloning, crossover and mutation over the existing population it will get the new population which may look like the ones given in Table B.3.

Table B.3: Example new population

10N+P/2K
N+P*K
2N/P+10K
7N-PK
4N*P-2K
N/5P/K

Finally after predefined number of generation it may possibly get to a perfect solution which satisfies all the records of the input dataset. Think that user has obtained “200N/P+100K” as the solution. Now this evaluable expression is similar to the mathematical formula which represents “Y = 200N/P+100K”. This resultant evaluable expression can take the input values of ‘N’, ‘P’, and ‘K’ and calculate the value of “Y”, which it gets as the output of the resultant evaluable expression.

Appendix C

How GPVLab Works

C.1 Introduction

In this section it is going to describe the GPVLab software in detail. An example scenario is taken to explain the process and capabilities of GPVLab.

C.2 How GPVLab Works

Consider the dataset shown in Table C.1.

Table C.1: Sample dataset

A	B	C	D
1	5	2	3
2	10	4	16
3	15	6	39
4	20	8	72
5	25	10	115
6	30	12	168
7	35	14	231
8	40	16	304
9	45	18	387
10	50	20	480

As you can see the above dataset has ten rows and it clearly has a pattern. Author has created this dataset with the intension to have the model "D = (A* B) – C" in it. Now it is going to present a step by step guide to discover a data model out of this sample dataset. Once GPVLab has been installed into a computer, it can be accessed via the GPVLab desktop icon (shown in Figure C.1) or through the 'Start > All Programs > GPVLab" menu. Screenshot of the GPVLab main window is shown in Figure C.2.

Figure C.1 – GPVLab Desktop Icon

As you can see, the main window contains a menu bar and a tool bar. First of all one should open a new explorer instance through the main window.

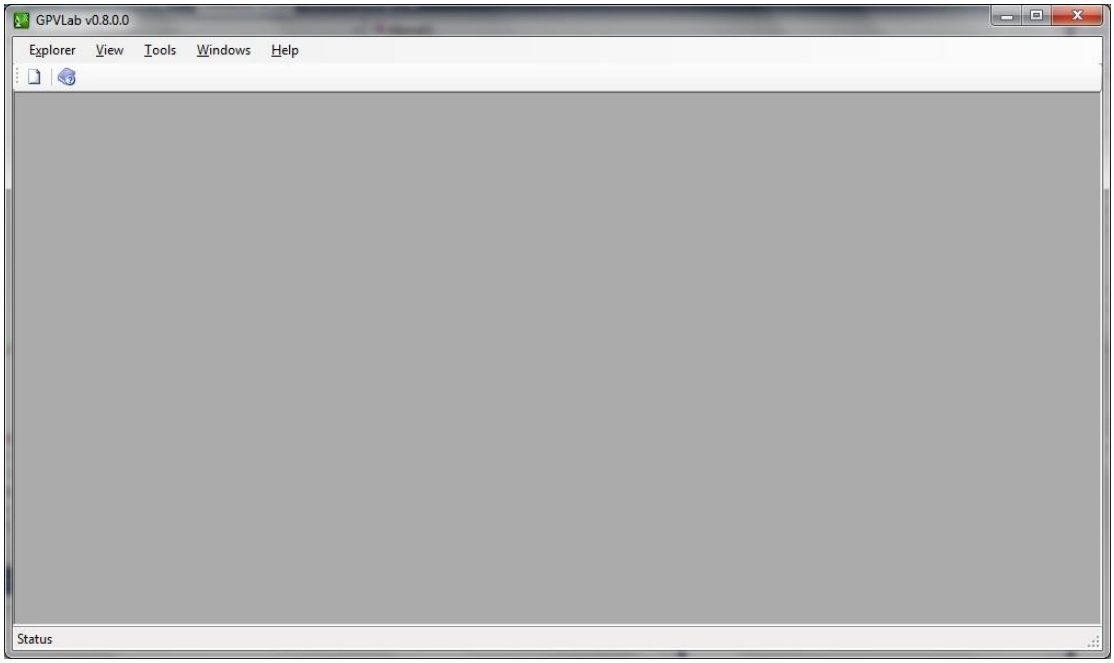

Figure C.2 – A screenshot of GPVLab main window

User can open a new explorer instance, either by clicking on the "New Explorer" icon (first icon in the tool bar) or through the GPVLab menu "Explorer > New". Inside the explorer window there are two ways to input data. Using "Data Input Methods", users can either create a dataset through GPVLab or export data from a comma separated value (.csv) file. If the user selects "Enter Dataset" then the system will prompt the user to enter the required number of columns for the dataset. Column names will be automatically generated. Once user enters a value and click "OK", the system will generate an empty dataset with defined number of columns. Now user can enter data into it using the space available at the bottom of the explorer window.

If user has a digital form of the dataset, it should be converted into a .csv file before exporting into GPVLab. Once the .csv file is in place, user can click on the "Upload csv File" button and select the required .csv file from the file system to import data into the GPVLab explorer window. Figure C.3 shows a screenshot of the explorer window. In this figure number '1' shows the "Data Input Method' Section. Number '2' shows the dataset area which shows the current dataset. This area is editable and users can add, modify or delete data from the dataset. The drop down list shown in '3' is the column name list, where user selects the reference column. Number '4' shows the "Advanced Settings" section. Number '5' and '6' shows the resultant expression in reverse polish notation (RPN) and infix notation, respectively. Number '7' shows

the visualization area where it shows the progress of the current model discovering process by the means of fitness of the best individual of each generation.

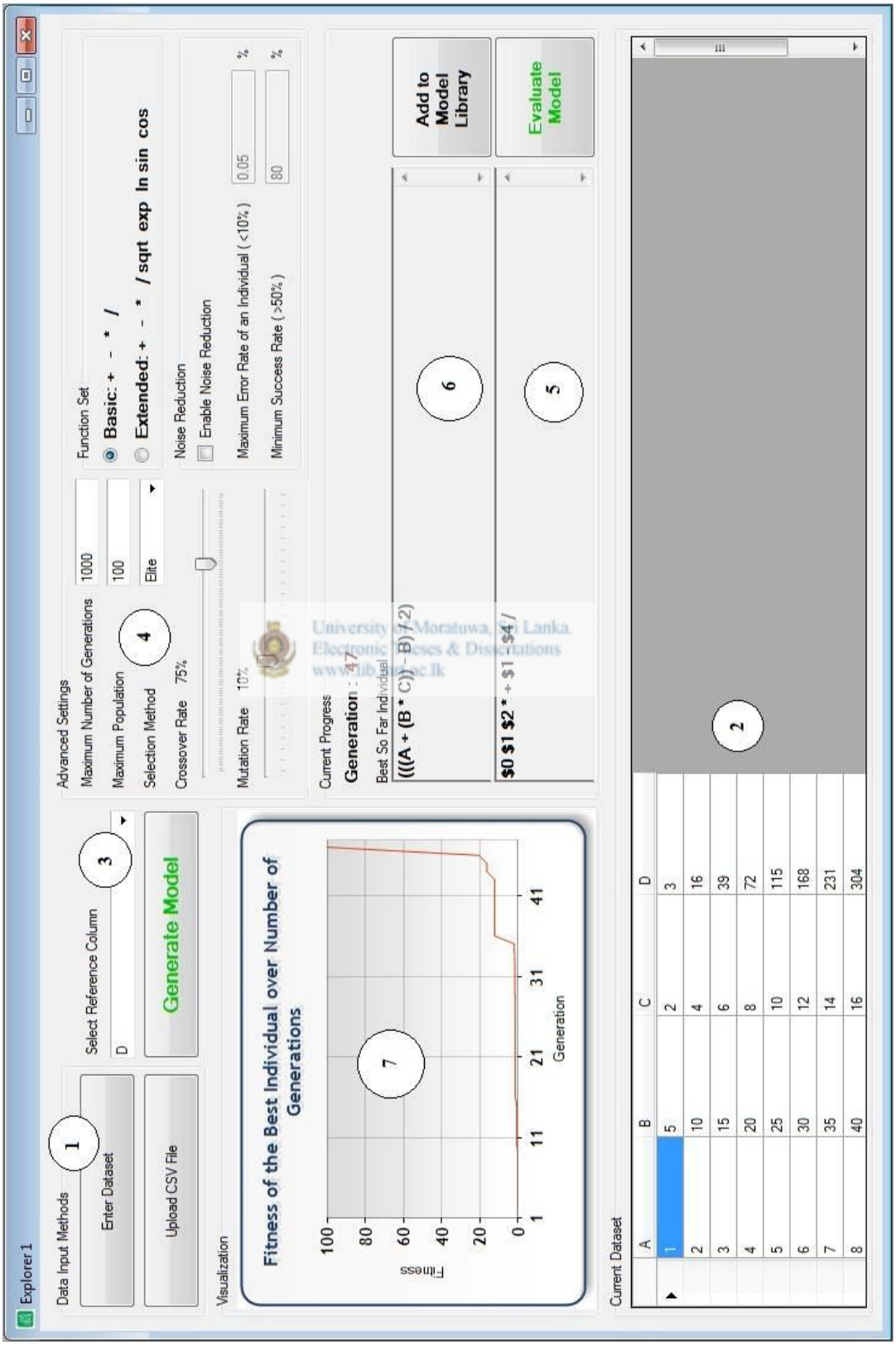

Figure C.3 – A screenshot of GPVLab explorer window

Once the user created or uploaded the dataset the next important thing is to select a reference column. This can be done through the reference column drop down list which lists down all the column names. Since the author has intentionally created this dataset to have the "D = (A* B) – C" data model, author has selected 'D' as the reference column. This means GPVLab will try to generate a data model using other columns which ultimately satisfy the data in the reference column. Next step is adjusting advanced settings. This is not a mandatory step, because default settings will be more than adequate for finding data models from most of the datasets.

Advanced settings section contains six major parameter settings.

- Maximum Number of Generations

  This is one of the termination criteria of the Genetic Programming process. Default value is '1000'.

- Maximum Number of Population

  Default value is '100'

- Selection method

  This contains three values, namely "Elite", "Rank" and "Roulette Wheel". Default value is "Elite".

- Function set

  GPVLab has an option to select one out of two function sets, namely basic and extended. Basic function set contains operators such as addition, substation, multiplication and division. Extended faction set has the ability to generate models consisting of sqrt (Square root), sin, cos, ln and exp (exponential) in addition to the basic operators. Please note that GPVLab can convert resultant evaluable expression in reverse polish notation (RPN), which only contains basic functions, into more human readable infix notation. Default is 'Basic'.

- Crossover rate

  Default value is '75%'

- Mutation rate.

  Default value is '10%'

Furthermore, if the dataset contains noisy data, users have the facility to enable noise reduction from the 'Advanced settings'. Once enabled, the system allows users to

change two important parameters. Those are the maximum error rate and the minimum success rate. Default values are '0.05%' and '80%', respectively.

After selecting the reference column and adjusting advanced settings user can initiate the model discovering process by clicking on the 'Generate Model' button. Users can see the progress of the discovering process through the visualization section which shows a dynamic graph of fitness of the best individual of each generation. Once the process is completed, either by exceeding the maximum number of generations or discovering a 100% accurate model, GPVLab presents the best expression as the result. For the example provided in this section, GPVLab has found a solution in just 47 generations. Nevertheless it was a different data model than expected. Interestingly this data model satisfied all the data records. GPVLab has found "((A + (B * C)) – B) / 2" as the solution. However the author was expecting "D = (A* B) – C" as the solution.

After obtaining the data model, users can directly evaluate it by clicking on the 'Evaluate Model' button. This opens up a dynamically generated window based on the expression. Inside this window users can enter values for each terminal of the output expression and get a result by evaluating the expression. Figure C.4 shows a screenshot of the dynamically generated "Evaluate Data Model" window for the resultant data model. In this dialog author has entered values of 5, 25 and 10 for A, B and C respectively. After successful evaluation GPVLab has obtained the value 115 which is the same as the respective reference column value. Now this model is ready to evaluate for the numbers which were not on the input dataset as well. This is so important in predicting next numbers of the sequence and evaluating for unknown numbers.

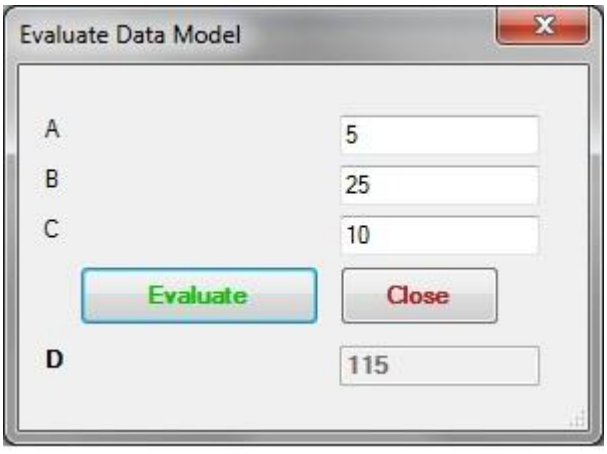

Figure C.4 – A screenshot of "Evaluate Data Model" dialog

Furthermore, if the user has decided to keep this model, there is an option to save the resultant model into the model library by clicking on "Add to Model Library" button in the explorer window. Once clicked, GPVLab prompts the user to provide a description about the data model before saving it to the library. Figure C.5 shows a screenshot of an 'Add to Library' dialog box.

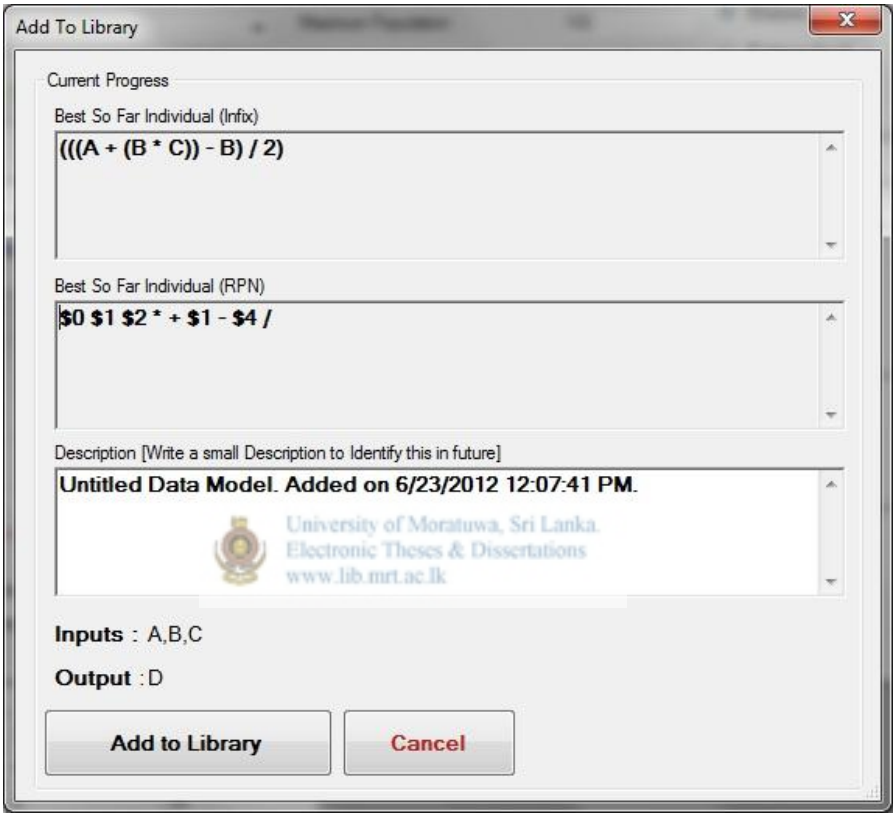

Figure C.5 – A screenshot of "Add to Library" dialog

Once the user has successfully saved a resultant expression into the library, it can be accessed anytime through the model library. Users can go to model library through the main menu "Tools > Library". Figure C.6 shows a screenshot of the Model Library with the model that has been discovered under this experiment. In the model library users can evaluate each model by clicking on respective "Evaluate Model" button. Users can remove unwanted obsolete models as well. Library contains a special feature, which helps users to export a data model as a standalone executable. Once user clicks on "Export to EXE", the system will open up a save as dialog to let the users to select a location and enter a name for the exe. Once selected GPVLab will automatically extract the data model from the library and generate a folder in the

selected location with all the necessary files including the standalone executable. Inside the folder, which has the same name as the executable name that user has provided, users can find the standalone executable file.

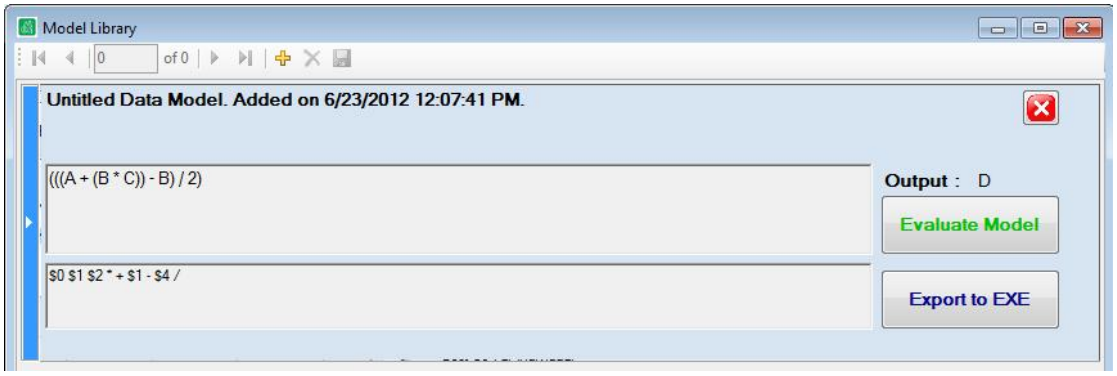

Figure C.6 – A screenshot of "Model Library" window

If user needs any help on GPVLab, there is a comprehensive integrated help file which can be opened at any time via the help icon on the tool bar. This can also be accessed via the menu "Help > Contents". Figure C.7 shows a screenshot of GPVLab's integrated help window.

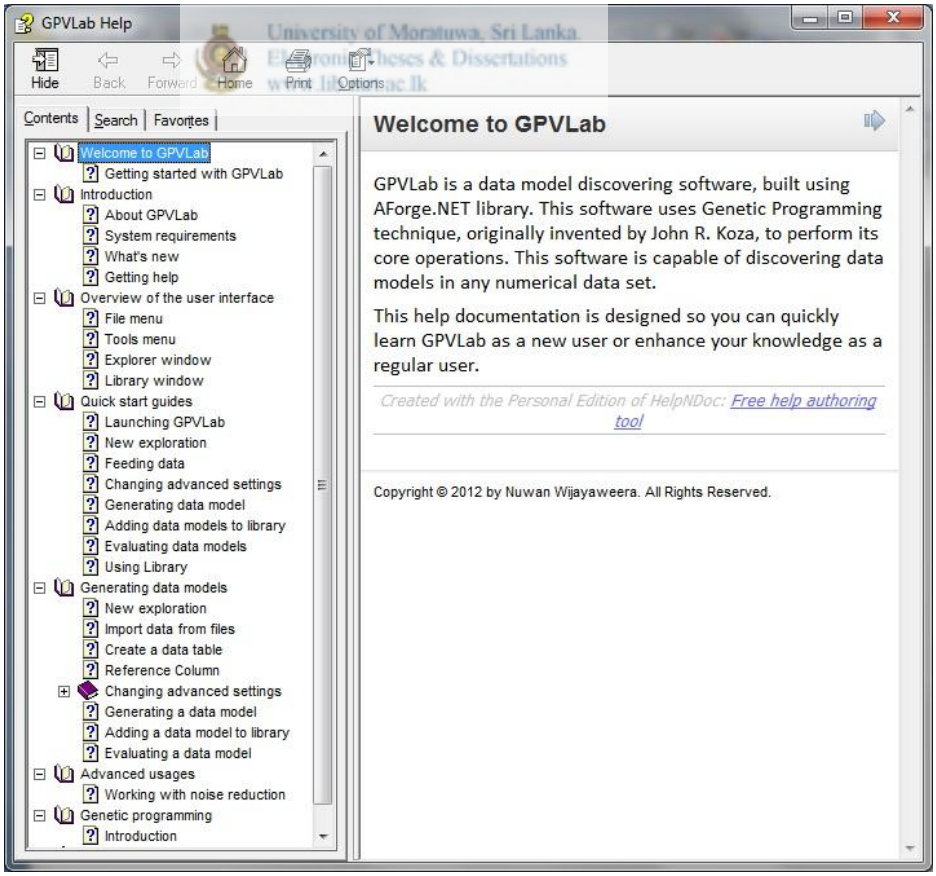

Figure C.7 – GPVLab integrated help window

Figure C.8 shows a screenshot of exported standalone executable and the folder which has been created after the exporting process. The “AForge.dll” which is inside that folder is a necessary DLL, which comes with the AForge.NET Framework. Now this folder is highly portable. Users can take this folder anywhere and the application inside this folder can be executed at any time on any computer which has .NET Framework 4.0 installed.

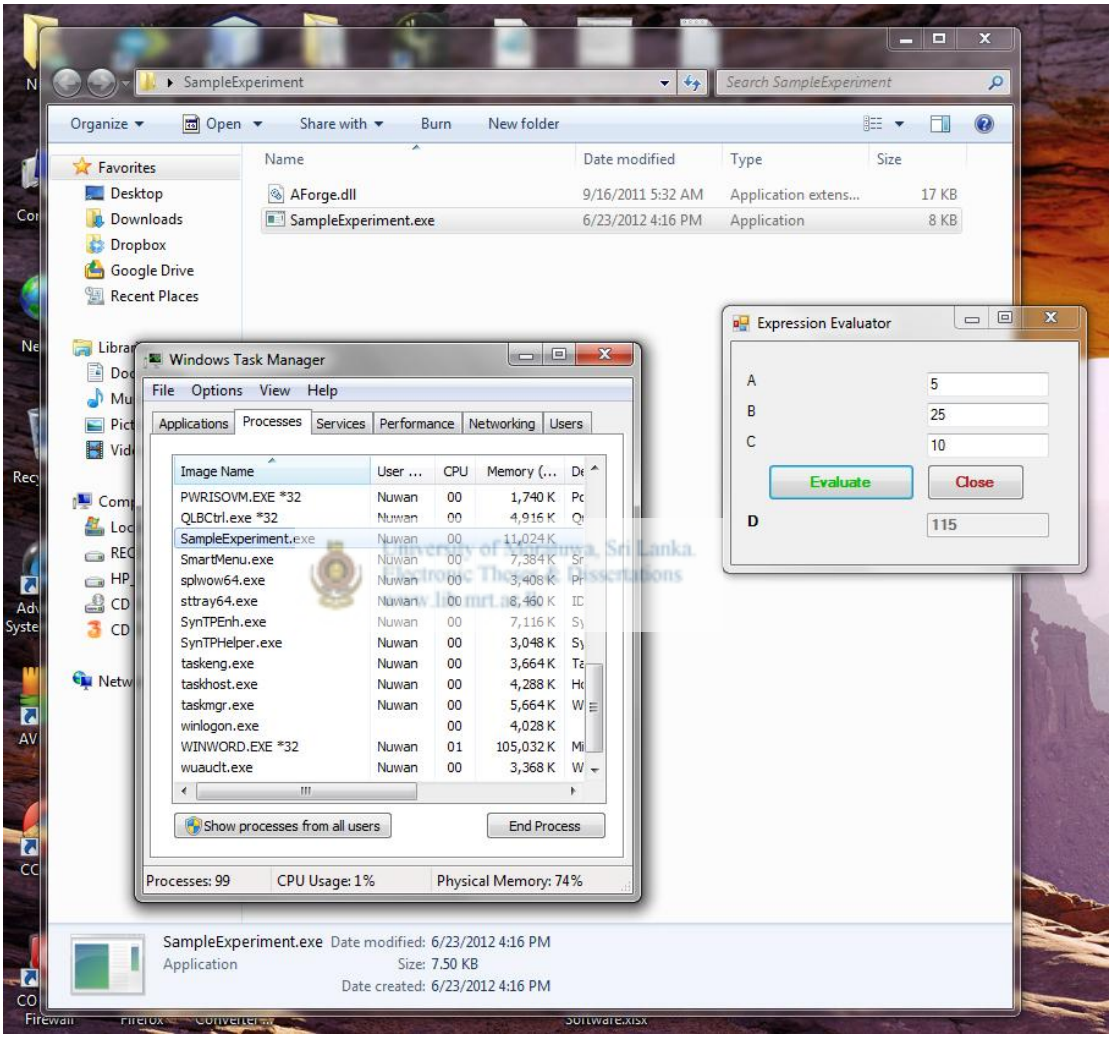

Figure C.8– Generated executable files and related screens

Appendix D

Main Dataset for Evaluation

D.1 Introduction

In this section it is going to present the data which author has taken for the main experiment. All the data were taken from the World Bank Data Catalog. This catalog contains data related to countries which were taken from World Bank datasets. This catalog contains various datasets under "Sri Lanka". All these datasets have important data collections related to Sri Lanka.

D.2 Main Dataset for Evaluation

For the purpose of the main experiment author has downloaded the data sheet under "World Development Indicators" section of Sri Lanka. This dataset contains a large collection of indicators and author has decided to take eight indicators from this. Author has taken "Total reserves (includes gold, current US$)" (**TR**), "Inflation from consumer prices (annual %)" (**I.CP**), "General government final consumption expenditure (% of GDP)" (**GGFCE**), "Exports of goods and services (current US$)" (**EOG.S**), "Gross domestic product (current US$)" (**GDP**), "Official exchange rate (LCU per US$, period average)" (**OER**), "Life expectancy at birth" (**LEABT**) and "Total population" (**PT**) and formed the dataset which is given under Table D.1. Please note that LCU means the Local Currency Unit.

Table D.1: Main Dataset for Evaluation (1960 – 2010)

Year	TR	I.CP	GGFCE	EOG.S	GDP	OER	LEABT	PT
1960	90000000	1.13444364	13.36763212	400490164.7	1454342131	4.761900004	58.20097561	10168000
1961	85000000	1.503579953	13.81374177	405386561.3	1458609620	4.761900004	58.54982927	10443000
1962	75000000	2.2729054	13.68384964	319741827.2	1241387714	4.761900004	58.92892683	10582000
1963	52000000	3.195647176	13.85840559	325377936.1	1321840726	4.761900004	59.3502439	10903000
1964	73000000	0.222783306	14.31222167	439915966.4	1698319328	4.761900004	59.82309756	11164000
1965	43000000	-0.15560166	13.88988845	391806722.7	1751470588	4.761900004	60.34978049	11439000

1966	55000000	2.189239332	13.72136771	380452674.9	1859465021	4.861105837	60.91353659	11703000
1967	52000000	5.860566449	13.13800504	371428571.4	1801176471	5.952370005	61.50109756	11992000
1968	40000000	7.456952734	12.5780248	361512605	1965546218	5.952370005	62.10285366	12252000
1969	42740000	5.866956078	11.8779274	584537815.1	2296470588	5.952370005	62.70878049	12514000
1970	50332571.3	2.665380209	12.55516014	583136593.6	2369308600	5.934948621	63.30995122	12690000
1971	59464516.69	6.34948605	12.44179183	570184254.6	2553936348	5.970317182	63.8984878	12861000
1972	86601280.49	9.626643102	10.9541404	700156250	2875625000	6.402499999	64.47041463	13091000
1973	77591857.49	12.3028868	11.53927054	944812030.1	3574586466	6.650749999	65.02268293	13284000
1974	57431421.39	6.625992553	9.331376754	1042225392	3791298146	7.007166666	65.56114634	13496000
1975	92329928.59	1.329518681	10.00231765	1043162901	3591319857	8.411999999	66.09865854	13717000
1976	292587699.3	1.224879588	8.564287088	1387936866	4104509583	8.872833333	66.64207317	13942000
1977	407115535.2	12.14159728	9.476151412	950352338.2	2733183857	15.61066667	67.18829268	14190000
1978	549102731.2	10.73192143	9.158760761	1134232498	3364611432	15.57183333	67.72478049	14472000
1979	282580139.5	26.1454101	8.545402618	1296672716	4024621900	16.53441667	68.22192683	14747000
1980	352359571.1	17.9689955	7.423092759	1345038961	4415844156	19.24575	68.64512195	14847000
1981	380230502.2	10.82574917	8.305286281	1304565113	4768765017	20.81225	68.972	15196000
1982	320947606	13.96438801	8.132334438	1360645984	5167913302	23.52858333	69.19482927	15417000
1983	530159878.2	16.63825375	7.762803585	1740762579	6043474843	25.43816667	69.31990244	15603000
1984	471725052.2	1.481180122	10.22263279	1555117820	5978460972	27.16258333	69.37273171	15842000
1985	377188000.9	7.976361936	10.29675608	1519200571	6405210564	28.01733333	69.39380488	16127000
1986	309506426.7	7.717165606	9.931731419	1683389946	6682167120	29.44475	69.422	16373000
1987	247726248.3	13.9915489	9.842689948	1819710783	6978371581	31.80675	69.48190244	16599000
1988	269432865.8	11.56753609	10.48469378	1904743412	6987267684	36.04708333	69.57804878	16825000
1989	447030482.6	21.49525205	9.759652438	2424288567	8032551173	40.06291667	69.67868293	17015000
1990	724220186.1	12.18563072	9.83845627	2586802030	9000362582	41.3715	69.73139024	17267000
1991	979864996.3	11.38343705	9.634055441	3082683094	9703011636	43.829625	69.69912195	17426000
1992	1653805525	11.74673702	9.166174572	3494577815	10338679636	48.3221675	69.57797561	17646000
1993	2069716975	8.448712487	9.670790421	3962059895	11717604209	49.41514167	69.39526829	17891000
1994	2111952732	7.674848734	11.47158012	4638263415	13029697561	51.25158917	69.22885366	18136000
1995	1984681138	15.93583104	10.5478514	4860502985	13897738375	55.27144417	69.1782439	18336000
1996	2042319918	9.573696264	10.35592453	5514307510	15091930836	58.994605	69.3162439	18568000

1997	1997795288	9.364243007	9.798268346	5724701319	15794972847	64.45011833	69.67714634	18784000
1998	1653728954	4.69170563	9.028413221	5555450170	15656342016	70.63545	70.25170732	19056000
1999	1131355393	6.17627591	10.51091344	6371581613	16330810304	77.00511667	70.98429268	19102000
2000	1357451877	14.1584558	10.26298513	5878254366	15746224410	89.38301333	71.78380488	18797000
2001	1705155579	9.55103167	12.71884438	5971095547	17102623876	95.662065	72.54390244	18921000
2002	2333781492	6.314637871	12.16054274	6543193121	18881765437	96.52095083	73.18156098	19173000
2003	2204869012	7.57592583	12.62979825	7300256942	20662525941	101.1944575	73.66514634	19435000
2004	2734967980	11.6396861	13.08868868	7892069652	24405791045	100.4980517	73.98934146	19644000
2005	2832098006	10.02018361	15.36193121	8516554444	28267410543	103.9144458	74.18263415	19858000
2006	3518418324	15.84211149	15.27221708	9419020069	32351184234	110.6232333	74.31207317	20039000
2007	2616559604	22.56449553	16.18316256	10114271208	40715249700	108.3337627	74.43326829	20217000
2008	5353610184	3.464963221	17.61111842	8972411693	42067965895	114.9447833	74.56660976	20450000
2009	7195358272	6.217648893	15.57091018	10746568194	49567521670	113.0644804	74.72260976	20653000
2010	90000000	1.13444364	13.36763212	400490164.7	1454342131	4.761900004	58.20097561	10168000

Appendix E

Dataset for Noise Reduction Experiment

E.1 Introduction

In this section it is going to present another dataset which author has taken for evaluation purposes. All the data were taken from the World Bank Data Catalog.

E.2 Dataset for Noise Reduction Experiment

Author has decided to take two indicators from the same dataset which was used for the main evaluation dataset. We have taken "Life expectancy at birth, total (in years)" and "Official exchange rate (LCU per US$, period average)" and formed the dataset which is given under Table E.1. Please note that LCU means the Local Currency Unit.

Table E.1: Life expectancy vs. exchange rate (1960 – 2010)

Year	Life expectancy at birth, total (in years)	Official exchange rate (LCU per US$, period average)
1960	57.86034	4.7619
1961	58.20098	4.7619
1962	58.54983	4.7619
1963	58.92893	4.7619
1964	59.35024	4.7619
1965	59.8231	4.7619
1966	60.34978	4.7619
1967	60.91354	4.861106
1968	61.5011	5.95237
1969	62.10285	5.95237
1970	62.70878	5.95237
1971	63.30995	5.934949

1972	63.89849	5.970317
1973	64.47041	6.4025
1974	65.02268	6.65075
1975	65.56115	7.007167
1976	66.09866	8.412
1977	66.64207	8.872833
1978	67.18829	15.61067
1979	67.72478	15.57183
1980	68.22193	16.53442
1981	68.64512	19.24575
1982	68.972	20.81225
1983	69.19483	23.52858
1984	69.3199	25.43817
1985	69.37273	27.16258
1986	69.3938	28.01733
1987	69.422	29.44475
1988	69.4819	31.80675
1989	69.57805	36.04708
1990	69.67868	40.06292
1991	69.73139	41.3715
1992	69.69912	43.82963
1993	69.57798	48.32217
1994	69.39527	49.41514
1995	69.22885	51.25159
1996	69.17824	55.27144
1997	69.31624	58.99461

1998	69.67715	64.45012
1999	70.25171	70.63545
2000	70.98429	77.00512
2001	71.7838	89.38301
2002	72.5439	95.66207
2003	73.18156	96.52095
2004	73.66515	101.1945
2005	73.98934	100.4981
2006	74.18263	103.9144
2007	74.31207	110.6232
2008	74.43327	108.3338
2009	74.56661	114.9448
2010	74.72261	113.0645