Model learning using genetic programming under full and partial system information conditions

Hassan Abdelbari [1], Sondoss Elsawah [1], Kamran Shafi [1]

[1] School of Engineering and Information Technology, University of New South Wales, Australia

[1] UNSW Australia at the Australian Defence Force Academy Northcott Drive, Canberra ACT 2600

hassan.abdelbari@student.adfa.edu.au, sondoss.elsawah@adfa.edu.au, kamran.shafi@adfa.edu.au

Abstract

Modelling is often an expensive and time-consuming process. Issues with modelling efficiency and implications to model adoption are long recognized in the broad modelling/simulation as well as system dynamics literature. Computational methods for supporting model learning provide an opportunity for addressing these limitations by providing computational tools that have the capabilities to analyse large datasets and explore possible model structures and parameters that can explain the behaviour of interest. This paper, a part of an ongoing research, aims to contribute to the area of using machine learning to support learning about system dynamics models. Towards this aim, the paper has three objectives. First, we present an overview of recent advances in system dynamics model learning. Secondly, we present and describe the proposed methodology which uses a genetic programming method under the assumptions of full and partial information available about the system. Thirdly, we present three case studies to develop and test the proposed methodology. Preliminary results show that the genetic programming was able to find good approximated models for first and second order system dynamics model. However, the results for third order models the learned models were not good approximations for the original models. Combining genetic programming and embedding reconstruction technique showed better results in the case of third order model.

1 Introduction

System dynamics modelling provides a set of conceptual and numerical modelling techniques that are used to help understand complex systems (e.g. socio-economic) (Meadows 1989). System dynamics is used to support policy analysis and design (Abdollahiasl, Kebriaeezadeh et al. 2014), learning (Bianchi and Bivona 2002), and

decision making (Ford and Sterman 1998). The system dynamics modelling includes four main phases: problem definition and formulation, conceptualizing the dynamic hypothesis, model formulation and testing, and model implementation.

Despite the value of system dynamics modelling in understanding complex systems and decision support, the adoption of system dynamics in deriving policy design has not reached its full potential yet. Issues with model building and use have been long recognized in the broader simulation modelling literature (e.g. (Robinson 2004), p16) and system dynamics literature in specific (e.g. (Winch and Arthur 2002)). First, the modelling process is often criticized for being time and resource consuming. The major bottlenecks in the modelling process are in the numerical model building phase and during experimenting with simulation models (Kanninga 2008). In a study of how experts allocated their time through the modelling process, (Willemain 1994) reported that 59% of the time was spent on actual model building, 16% on model evaluation, 14% on problem definition, 9% on fitting model parameters, and only 2% on model implementation. Improving the efficiency of the modelling process carries the opportunity of allowing modellers the time to concentrate on other high-impact modelling activities, such as understanding views and assumptions underpinning models, and deriving insightful recommendations about the problem behaviour. Second, the numerical system dynamic model is developed to test a priori defined problem structure (known as dynamic hypothesis). The user can run experiments to change model parameters and observes changes in the systems performance. The rigid structure is limiting in situations where there is uncertainty about the problem structure and/or the decision maker is interested in exploring the system behaviour under a wide range of system designs and policy variants. To examine an alternative structure, this means redeveloping the model, which comes with additional costs and time. Finally, these challenges are exacerbated by the rapid increase in large data sets (i.e. 'big data') which brings the need to analyse data in an efficient and useful way to support model formulation and calibration (Pruyt, Cunningham et al. 2014).

Computational techniques may be used to overcome these limitations and support the modelling process by providing the modeller with computational frameworks and methods that have the capability to analyse the data, and explore possible model structures and parameter that can explain the behaviour of interest.

This paper aims to contribute to the area of using computational intelligence method called genetic programming in learning system dynamics model in two different scenarios: full and partial knowledge of system information and examine how these two conditions affect the results. Toward this aim, the paper is organized around the following objectives:

1. Present an overview of recent work in learning system dynamics models using computational methods (Section 2)

2. Present a proposed methodology based on genetic programming for model learning (Section 3)
3. Use three case studies to apply and test the proposed methodology (Section 4 and 5)

2 Research Background

We mean here by model learning is the iterative process of exploring possible structures (i.e. assumptions about the key variables and causal relationships underpinning the model) and estimating model parameters which best explain the behaviour of interest. Some efforts have been made towards model learning for system dynamics models using computational methods. In this section, we discuss some of these efforts along with their relevance to our work.

(Medina-Borja and Pasupathy 2007) used classification and regression trees (CART) and Chi-square interaction detection (CHAID) to find the causal relations among the variables and based on these causal relations, structural equation modelling (SEM) method is applied to transform these relationships into mathematical equations. The authors assume that the modeller does not have any support from system experts, but just relies on large amount of information stored about the system. This work is similar to our proposed methodology in finding the mathematical structure between variables but the method of structural equation modelling is done in manual fashion by the modeller by making assumptions about the mathematical relations and evaluate these relations by comparing their behaviour with the actual one and repeat this process until find the best equations.

Recurrent neural networks combined with genetic algorithm are used by (Jeng, Chen et al. 2006) and (Chen, Tu et al. 2011) but for different purposes. (Jeng, Chen et al. 2006) converted the stock and flow diagram to its corresponding recurrent neural network by representing each level variable by tuple of input, output and state units, each rate variable as a hidden unit and each constant as a weight parameter. Then, genetic algorithm is used to train the network by adjusting the weights and finally the network is converted back to the stock and flow diagram with the estimated values for the parameters. In this work it is assumed that the model structure is known and the genetic algorithm is used for parameters estimation. Similar to this work, (Chen, Tu et al. 2011) uses the same steps but add to it the ability to evolve the structure of the neural network using the genetic algorithm by adding or removing links or units simultaneously with parameters estimation. This work is very close to our proposed methodology in attempting to find the structure and the parameters simultaneously, but the step of converting the network back and forth into stock and flow diagrams is adding additional computations to the learning process.

(Önsel, Önsel et al. 2013) used the agglomerative hierarchical clustering algorithm to identify the mode of behaviour that can explain the behaviour of interest (i.e. exponential growth, s-shaped, oscillation) by using different features to represent the time series curves. Based on the clustering results the pre-defined model parameters are estimated based on the type of the output mode. In this work the model structure must be given before parameters estimation and this is not the case with our methodology that is able to find both the structure and parameters simultaneously without any prior given structure.

(Takahashi 2006) used the predicate logic to convert the causal diagrams into stock and flow diagrams based on defined logical rules.

Evolutionary methods are commonly used for learning both the model structure and estimating the parameters or just estimating the parameters. For parameters estimation, genetic algorithm (Yücel and Barlas 2011) (Chen and Jeng 2009) (Grossmann 2002) (Duggan 2008), co-evolution genetic algorithm (Liu, Howley et al. 2012), simulated annealing (Graham and Ariza 2003) and particle swarm optimization (Liu 2012) are used by assuming that the model structure is already given and they are used to estimate the model parameters until the error between the actual output and the model output is minimized. This approach is different from our approach because the proposed methodology is not requiring any prior structure to be given, but it able to find both structure and parameters at the same step. For learning both the structure and parameters simultaneously, genetic programming algorithm is used to find the mathematical structure of the model to be used within a decision support system used for forecasting in the work proposed by (Pawlas and Zall 2012). (Geyer-Schulz 1996) used genetic programming to find the model for the beer game by introducing another version of it by finding the fuzzy rules instead of the mathematical equations. This line of work is relevant to our work in terms of attempting to find both the structure and parameters simultaneously, but our methodology goes further by making different assumptions about the extent of information available about the model. The first one is the full information condition where we know number of outputs and their observations and parameters ranges but not the number of parameters and the second one is the partial information condition where we are not given anything except one observation for one output variable and we try to reconstruct other observations if there is any to use them in model learning.

Our literature review of this area reveals two gaps. First, most of the methods for learning the models are assuming the existence of the model structure and the rest of the task is reformulated in model parameters estimation. Second, the methods that are considering the learning of both model structure and parameters estimation needs pre-defined assumptions to be used such as the ranges of the parameters, number of parameters and the number of observations. This ongoing work aims to address these gaps by applying the genetic programming algorithm to find the model differential

equations for each stock variable and estimate the model parameters simultaneously with two different scenarios:

- The first scenario assumes that full information about the system is provided as well as the system observations (e.g. number of stocks and their observations, the possible ranges the model constants could have and what type of mathematical operations could be used by the genetic programming algorithm.)
- The second scenario assumes that partial information is provided (i.e. access to observed data for one stock) and no other information is available like in the case of full information. In this step the range of constants will be assumed to be within 0 and 1 and the range of mathematical operations will combine both linear and nonlinear relations and finally based on these assumptions, nonlinear analysis method will be used to reconstruct another hidden observations of the system based on the given one which could have an effect on the observed patterns.

3 Proposed methodology

3.1 Overview

Although system dynamics approach is computational in nature itself, a crucial stage in system dynamics modelling, which requires significant input from human modeller, is the problem formulation and conceptualization of dynamic hypothesis about the model. Hence, computational, specifically computational intelligence, techniques have the potential to provide modelling support and automate these processes.

At a higher level this stage involves two tasks:

- Identifying and mapping the variables (stocks, flows, state variables, etc.) to represent all factors of interest in the underlying system or part of the system to be modelled; and
- Identifying and formulating the mathematical relationships between these variables

The first task is highly qualitative and human-driven (Wolstenholme 1999), so it is relatively more difficult to be represented as a task to be solved by machine intelligence. Conversely, the second task is more tedious for human modeller due to its analytic but relatively easier to be represented as a computational task. As discussed in Section 2, there is overall limited work in automating the modelling process from this angle. Within the available techniques most focus on the second task of finding the relationships between variables, fewer that focus on the first task and probably none (to the best of our knowledge) that tackle both tasks together in an integrated fashion to provide modelling support as discussed above. The main motive of the methodology

proposed in this paper is to address this gap using computational intelligence techniques.

To highlight the novel features of our proposed methodology, we summarise the generic steps used by the modelling support techniques:

1. Obtaining or recording dynamic system behaviour in terms of time-series, transactional or other forms of data representations;
2. Identifying key components or variables of the target model contributing to the recorded system behaviour in step 1;
    a. Known model components approach: assumes variables of interest including rates and their ranges, stocks and other state variables are known and are provided by the modeller.
    b. Unknown model components approach: assumes the variables of interest are not known and need to be identified or constructed from the given observations;
3. Model (or model structure) learning using the identified variables in step 2 through following the process in 2a or 2b.
4. Calibrating, evolving and validating the learnt model using the observed system behaviour and the intermediate models.

Several iterations between 3 and 4 are usually required to obtain a fine tune model that closely matches the observed behaviour.

As discussed in Section 2, most extant modelling support approaches focus on learning model structure (step 3) while implicitly relying on known information about model components (2a). In addition to provide modelling support for step 3 above, the techniques proposed in this paper explicitly considers the modelling support needed in step 2 of the generic process above and specifically apply techniques that cater for the situations where model components are not known a priori.

Specifically, the proposed methodology uses a genetic programming (GP) approach for model learning and couples this with a time-delay embedding reconstruction (TDER) method to identify key variables under unknown or incomplete information scenario. To compare the performance of this additional automation step, the GP is also used to learn model under known or complete information scenario. Although we note that even in complete information scenario, our GP based approach does not assume knowledge of the number of flows or rates. Figure 1 graphically depicts this methodology using a flow diagram notation. The details of two computational techniques i.e. GP and TDER are discussed in the subsections below.

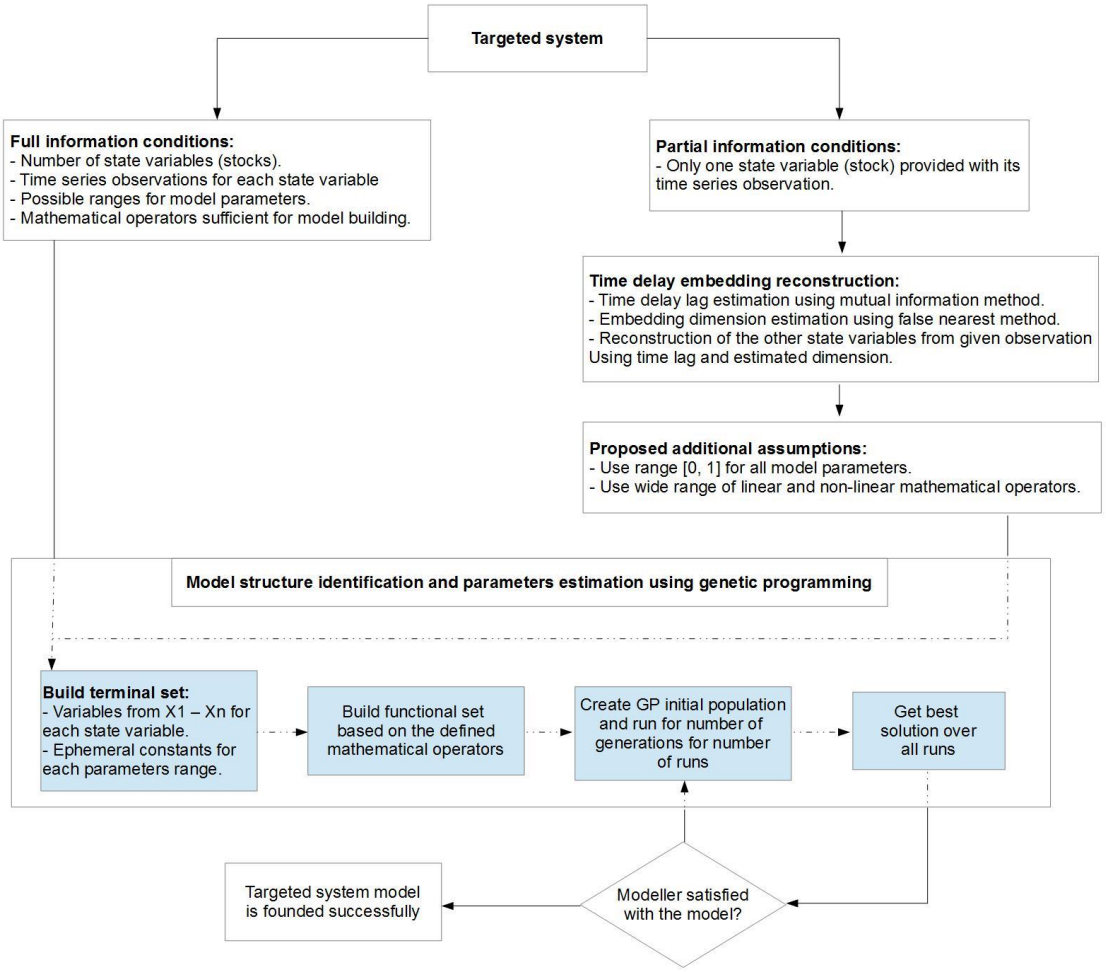

Figure 1: The proposed methodology

3.2 Genetic programming

Genetic programming is an evolutionary computational method like genetic algorithm but the individuals in the population are represented as programs in tree structures and this population of programs are evolved over time to produce another generations of programs using genetic operators such as crossover and mutation (Langdon, Poli et al. 2008). In order to use genetic programming to solve problems such as finding the differential equations for a system based on the observations, several preparatory steps must be followed. The first step is defining both the terminal and functional sets. Due to the tree representation of the individuals in genetic programming, those sets are used to specify the types of nodes that will be used to build such trees. Terminal set contains the types of nodes in the tree leaves and this set can contain independent variables, constants and constants with random values. In addition to this set, the function set is responsible for defining the types of internal nodes of the tree that define the mathematical or logical operators that will be used on the terminals and this set can

contains the algebraic operations such as plus and minus and not linear functions such as sine and log. In addition to the mathematical functions, function set can contains logical operators such as AND, OR and NOT. After defining these two sets, the fitness function is defined to assign fitness value for each individual. After that, the parameters that control the algorithm runs are defined and the final step is defining stopping criteria and this can be achieved through reaching the maximum number of generations or finding the optimal solution (Poli, Langdon et al. 2008).

In order to make the explained concepts mentioned earlier related to the genetic programming algorithm clearer, a simple example is provided. Figure 2 shows a simple tree representation for a program min(x*y, 2.5+x-y). To generate such program the terminal set should contain both the variables x and y, in addition to a random variable that responsible for generating random numbers that assigned to some nodes at the first step of creating the initial population and these values are not changed during the algorithm execution. The function set here may include both algebraic operators and min and max functions.

Based on terminal and functional sets, the initial random trees are generated by starting with random root node from the functional set and at each step another random nodes are selected from both sets until the reaching the tree initial maximum depth. Three main methods are used to generate the initial population, full method, grow method and ramped half and half method. In the full method all the nodes in all depths are selected from the functional set until reaching the maximum depth which filled with nodes from the terminal set. In grow method, only the root node must be from the functional set and all other nodes in each depth could be from either the functional or terminal set. In the ramped half and half method, half of the population are created using the full method and the other half generated using the grow method. In each half an equal number of trees is generated with different depths range from 2 to the maximum initial depth.

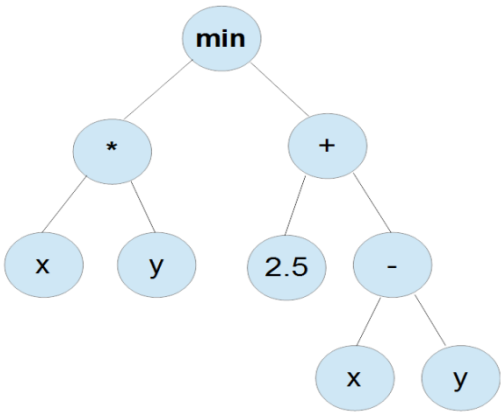

Figure 2: GP tree representing the program min(x*y, 2.5+x-y)

After generating the initial population, the fitness value is calculated for solutions. Based on these values a selection method is applied to select the candidate solutions that

will be used for matting in order to produce the next generation. The main mating operators are reproduction, crossover and mutation. In reproduction, the selected solution is passed to the next generation without alternation. In crossover, two parents are selected and for each parent a crossover point is selected and swapping the corresponding subtrees. In the mutation operator, a mutation point is selected and the subtree at this point is replaced with random generated sub-tree. This process is repeated until the stopping criteria are met. Figure 3 shows an example of applying the crossover operator and Figure 4 shows an example of applying mutation operator on selected parents in order to produce the off-springs.

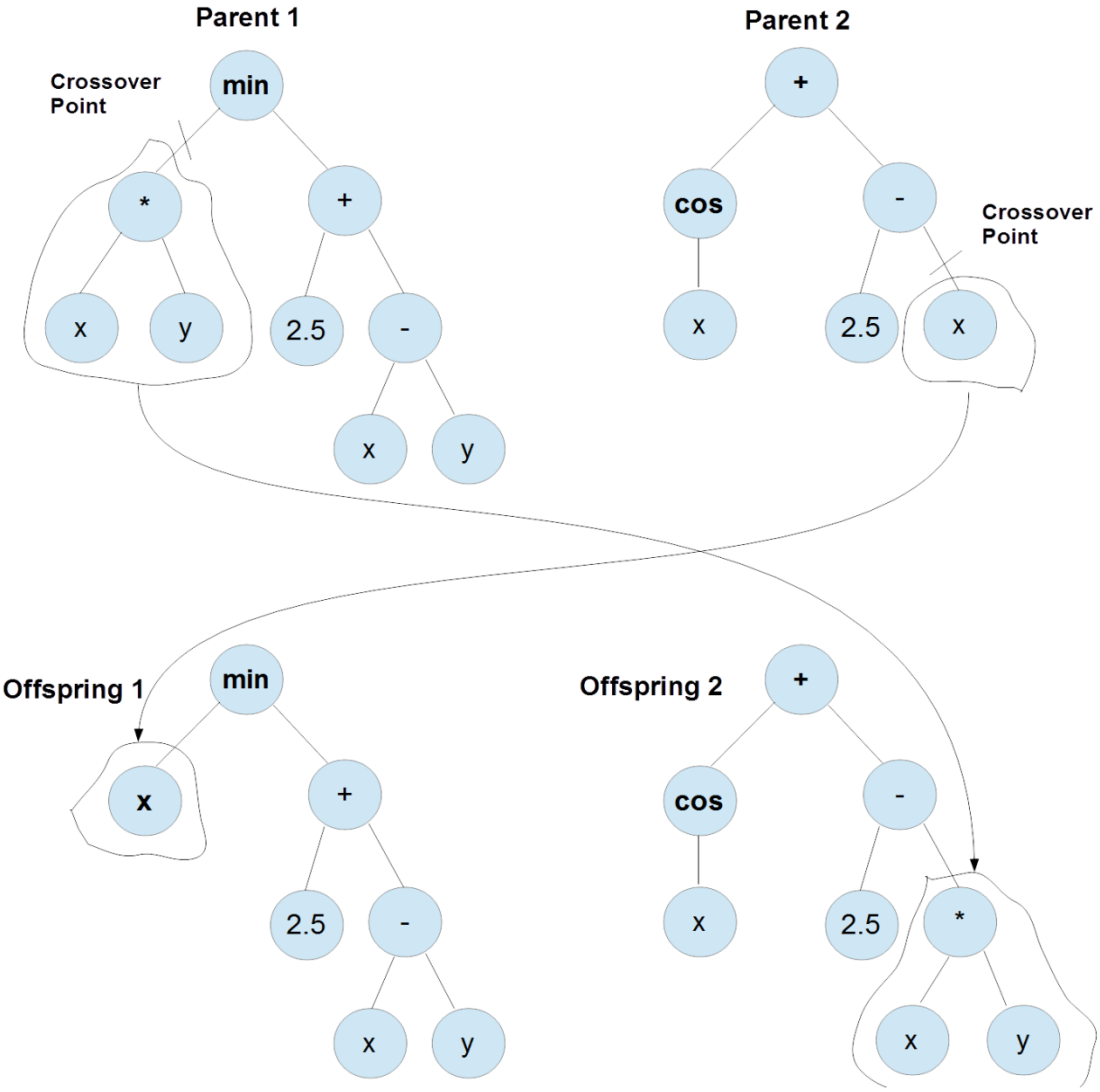

Figure 3: Crossover operator example

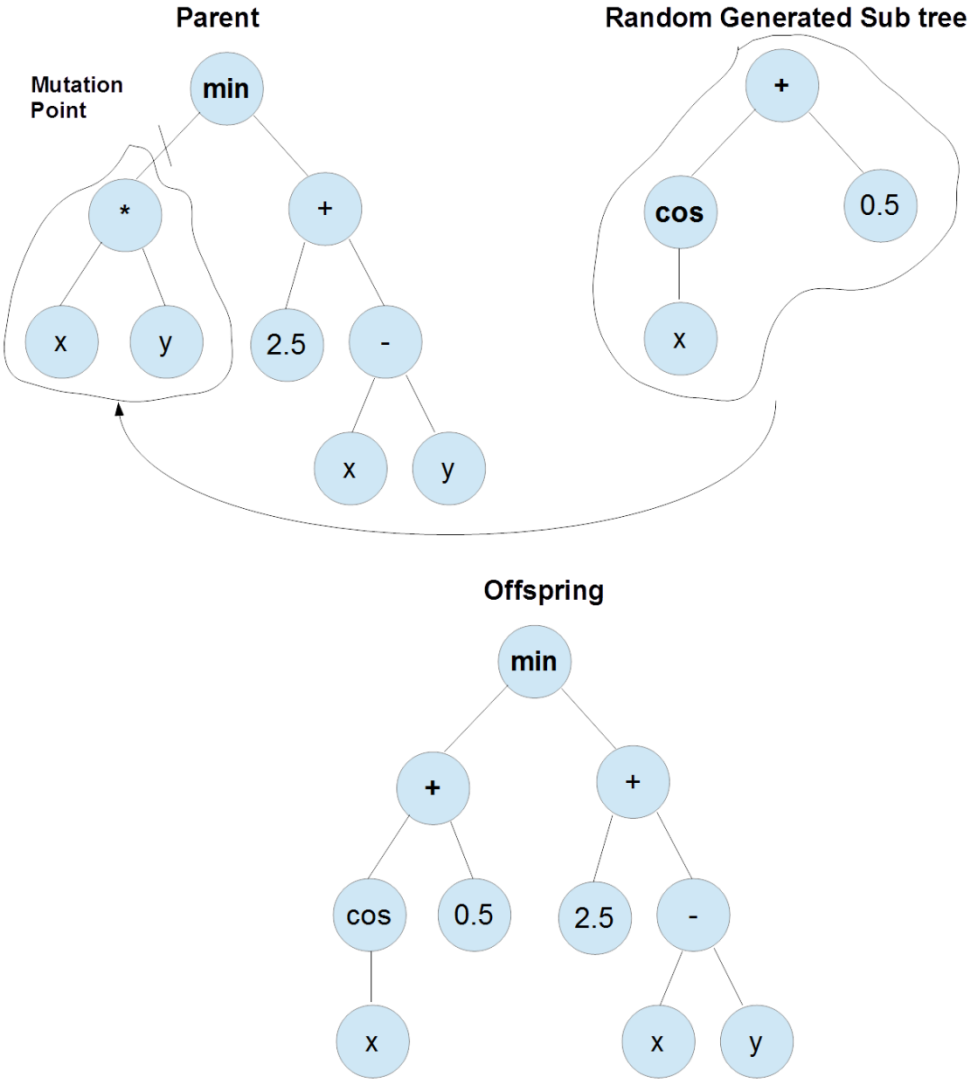

Figure 4: Mutation operator example

Genetic programming in its normal version described above used to find the mathematical equation for one variable since that each candidate solution is represented by a tree describes the equation that control the behaviour of this variable, but if the problem is to find a set of mathematical equations for several variables, another version of it called multi gene genetic programming is used by constructing solutions that represented as group of subtrees and each subtree describes one equation for each variable and all these trees are grouped under one root (Garg, Garg et al. 2014). In this version, the solution is composed of more than one tree, one for each independent variable.

The fitness function that will be used in the proposed implementation is the complexity invariant distance which is a Euclidean distance added to it a factor of measuring the

shape of the time series (Batista, Wang et al. 2011). The procedure of calculating the fitness function for each solution is given by the following steps:

- *Step1:* Retrieve the actual observations for stock variables.
- *Step2:* Use each stock observation initial value for the initial value for the generated solutions simulation.
- *Step3:* Run a simulation model for each solution based on the equation and initial values for stocks to generate a time series data for each stock variable.
- *Step4:* Use the distance measure to calculate how close the actual observations to the simulated observations are for each stock variable.
- *Step5:* Sum all these distance values and this will be the value of the raw fitness for the solution.
- *Step6:* Calculate the adjusted fitness from the raw fitness.

The genetic programming algorithm has some control parameters that need to be set before running the algorithm. These parameters are represented in, population size, generations number, the function set, terminal set, crossover rate, mutation rate, reproduction rate, probability of selecting internal nodes for crossover points, probability of mutation for each node, initial maximum depth, maximum depth, initial population generation method, crossover operator, mutation operator and selection method. In order to not get lost of our main focus here in using genetic programming in finding the underlying model for a system based on the system observation, we will not try to find the best value for each parameter and we will just rely on the literature of the genetic programming field in defining the commonly used ones. These values will be fixed for all runs with some variations. There are two main parameters that considered as problem dependent and they are the terminal set and function set. These configurations will be defined for each experimental runs on each example. The other parameters will be defined from the literature.

Based on the results conducted on both (de Lima, Pappa et al. 2010) and (Feldt and Nordin 2000), they found that the main control parameters that affect the results of the genetic programming are the population size and number of generations. The population size used in these experiments range from 50 to 2000 and number of generations are range from 50 to 250. The other values of the pavements are commonly used from Koza book about genetic programming (Koza 1992) which use population size of 500, number of generations of 51, crossover rate of 0.9, reproduction of 0.1, initial depth size of 6, maximum depth size of 17, fitness proportionate selection method, ramped half and half method for generating initial population, probability of selecting internal nodes a s crossover point of 0.9 and no mutation operator is applied.

3.3 Time delay embedding reconstruction

Taken theorem, from nonlinear time series analysis, states that if an observation from a system is provided, other unmeasurable observations can be reconstructed from this observation in terms of the assumption that this observation from the system is a result from projecting the other system observations into less dimensions (Takens 1981).

In order to apply the reconstruction on system observations two parameters must be found, time lag and embedding dimension. Several methods are used to estimate both of these parameters separately or simultaneously. The common used method to estimate the time lag parameter is the mutual information method (Fraser and Swinney 1986). In this method a time lag is selected in range from 1 to specific number and fixing the embedding dimension to be 2 and for each time lag the mutual information between the original time series and itself delayed by time lag is calculated. The best time lag value is the value where the mutual information curve has its first minimum.

After estimating the time lag the embedding dimension is estimated using a method called false nearest method (Kennel, Brown et al. 1992). The idea behind this method is starting by increasing the dimension and calculates the ration of true neighbours to false neighbours, if this ratio is less than specific threshold then the first value after this threshold is the best value for the embedding dimension. False neighbours means that if two points at the current dimension are no longer neighbours in the higher dimension then they are considered as false neighbours.

After estimating both time lag and embedding dimension the other hidden observations are reconstructed from the original observations. This step of time delay embedding reconstruction which includes the estimation of time lag, estimation of embedding dimension and the reconstruction are performed in using time series analysis package called TISEAN (Hegger, Kantz et al. 1999).

4 Experimental setup

4.1 Case studies

To evaluate our implementation for the proposed methodology in finding the dynamic hypothesis underlying the system dynamics model, we use three case studies of system dynamics models. These models are simulated and the simulated data are used to find the underlying model that responsible for generating such behaviour using the proposed methodology. The models that are founded are compared with the original models in order to validate how accurate the proposed methodology is. These case studies selected to have a variety in models orders. The first case study has one stock, the second one has two stocks and the last case has three stocks. We selected these varieties to measure how the GP will perform when the model become more complicated with more than one observation those are needed to be optimized. The first two case studies are selected from system dynamics self-study course at MIT open course ware and the third case study selected from (Sterman 2000).

4.1.1 Case 1: Balance in bank account model

Figure 5 shows the stock and flow diagram for this case. The balance in bank account case is describing how the amount of money in bank account is changing over time based on the rates that affecting this behaviour.

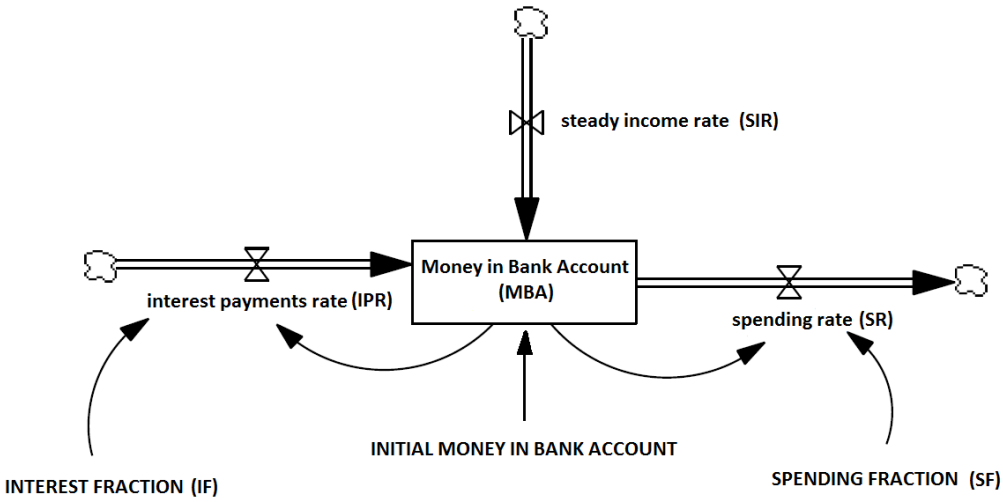

Figure 5: Bank in account model stock and flow diagram

4.1.2 Case 2: Epidemics model

The epidemics case describes two coupled stocks, where the first stock represents the number of uninfected people and the other stock represents the sick people. This is more complicated model than the previous one to test the ability of GP to find the differential equations for two stocks that are affecting each other. Figure 6 shows the stock and flow diagram for this case.

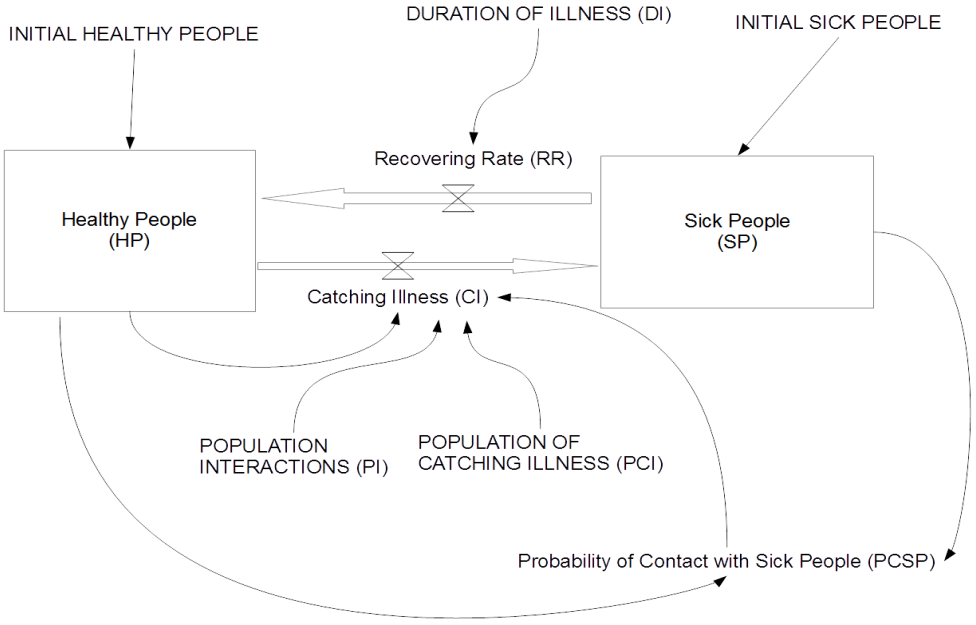

Figure 6: Epidemics model of healthy and sick people stock and flow diagram (Figure 35, Page 90, An Introduction to Sensitivity Analysis Chapter, Prepared by Lucia Breierova and Mark Choudhari Spetember 6, 1996, Under Supervision of Dr. Jay W. Forrester for MIT System Dynamics in Education Project, System Dynamics Self-Study Course at MIT Open courseware)

4.1.3 Case 3: SIR (Susceptible Infected Recovered) model

The SIR case describes how three stocks are changing over time based on the rates that affecting their behaviour. The first stock variable represents the susceptible population, second stock represents the infected population and the last stock represents the recovered people. This is similar to the epidemics model. Figure 7 shows the stock and flow diagram for the model.

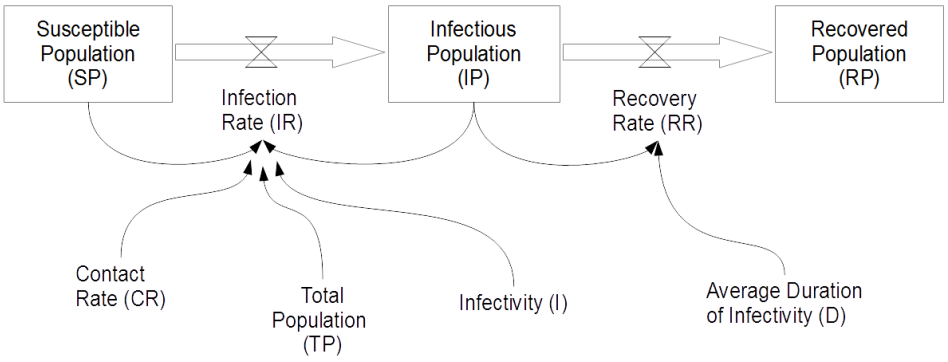

Figure 7: SIR stock and flow diagram (Figure 9-5, Page 304, Part 3, The Dynamics of Growth, Business Dynamics, Systems Thinking and Modeling for a Complex World, John D. Sterman, 2000)

4.2 Setting the GP parameters

Table 1 (Appendix A) shows the parameters settings that will be used to run the GP algorithm and for each case study the dependent configurations will be defined. For any more detailed reading about these parameters, their definitions and effects on the GP performance can be found in (Koza 1992). As discussed the GP algorithm will be used to find the differential equations for system dynamics model based on the stocks observations.

5 Results

In this section, we show the results of employing the GP algorithm to find the equations of system dynamics models in all cases with full and partial information conditions. For each case, the behaviour of the solution obtained by GP is compared with the actual behaviour of the model, and the solution tree structure is presented. The parameters used for each run and the fitness values over both populations and generations can be found in Appendix B.

5.1 Case 1

5.1.1 Full information conditions

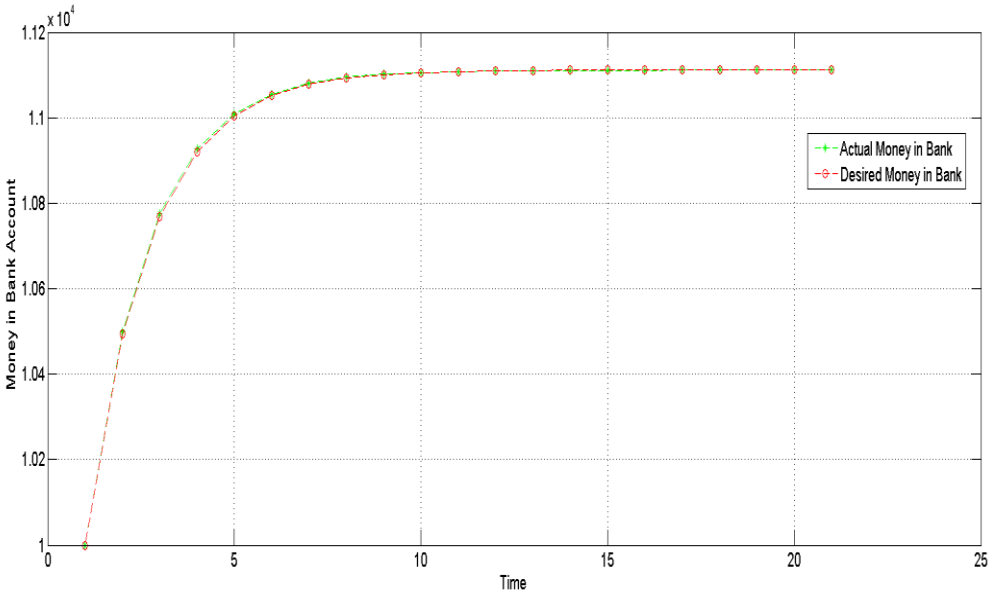

Figure 8: Best solution money in bank compared with the actual money in bank

From the conducted results we can see that GP provides good results in the case 1 of one stock variable. As seen from Figure 8 the desired and actual values for money in bank are very close to each other. From the structure perspective, the differential equations structures for both the actual model and the model that GP found are very similar as the following:

Actual targeted equation:

$$5000 - 0.45\ MBA$$

Best solution equation:

$$4930.24 - 0.44\ MBA$$

5.1.2 Partial information conditions

In this experiment the reconstruction is accomplished by assuming that we have only the money in bank observation and apply on it the reconstruction steps to produce other observations, the following steps describes how this steps are accomplished in selecting the time lag and embedding dimension. We apply this step to see if the GP will produce similar structure by applying GP in the original observations.

5.1.2.1 Estimating time lag

Mutual information method is applied for estimating the time lag and the best value found is 1, Figure 9 shows the mutual information curve over different time lags.

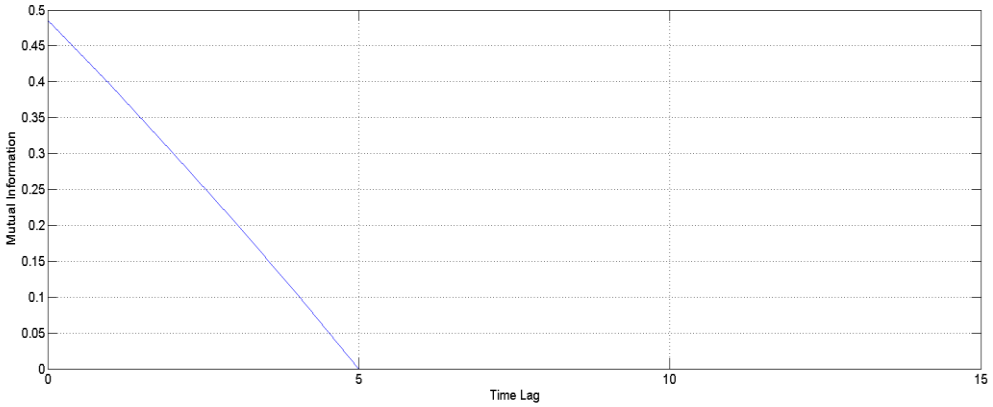

Figure 9: Mutual information over different time lags

5.1.2.2 Estimating embedding dimension

False nearest method is applied for estimating the embedding dimension and the best value found is 1 by using threshold value of 0.1, Figure 10 shows the false nearest curve over different embedding dimensions.

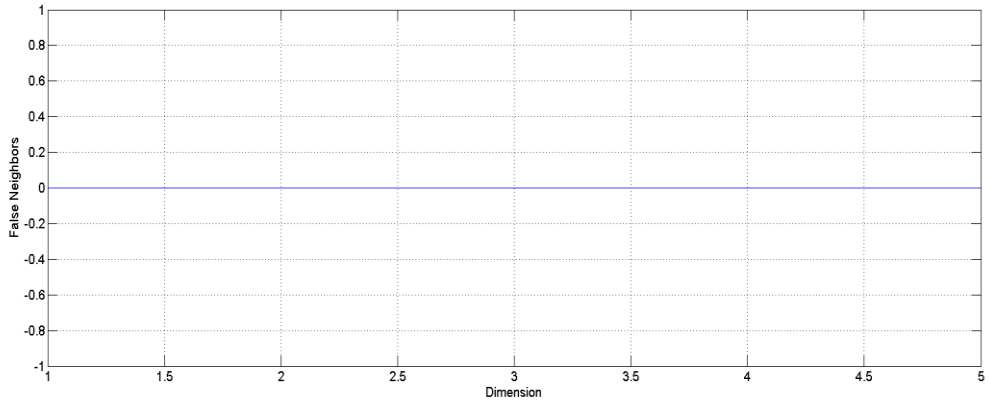

Figure 10: False neighbours over different dimensions

5.1.2.3 Reconstruction

Since the estimated dimension here is one, the reconstruction step tells us that there are no other observations involved, so the actual observation of money of bank that is produced from the model is the only one. This observation will be used with the other configurations of assuming the range of constants is all within 0 and 1 and we use a combination of linear and nonlinear operations.

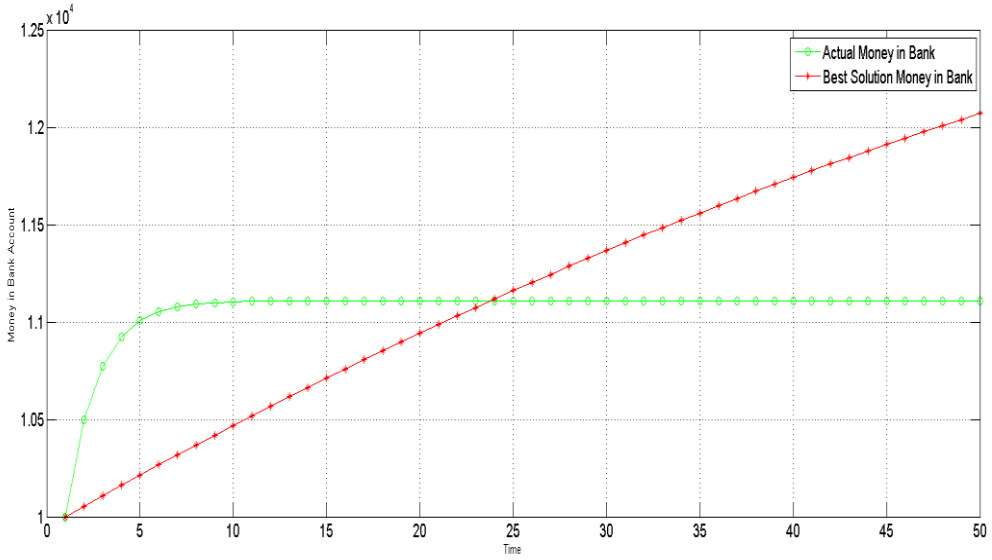

Figure 11: Best solution money in bank compared with actual money in bank

As we can see from the Figure 11 that the best solution output is seems linear and not following the actual output behaviour in the case of partial information. This behaviour may due to the absence of the parameters ranges that are define in the full information case, but here we use the range between 0 and 1 for all parameters values. If we compare the equations structures of the best solution and the actual model equations they will be different such as the following:

Actual equation:

$$5000 - 0.45\,MBA$$

Best solution equation:

$$85.19 - \ exp^{0.05* \sqrt[2]{MBA}} * 0.013 * \min(MBA, \max(0.42, MBA)$$

The best solution equation contains more complicated terms and this is due to the use of a function set of both algebraic operations and some nonlinear functions and this also due to the absence of any prior information provided about the minimum mathematical operators that could be used to find the equations.

5.2 Case 2

5.2.1 Full information conditions

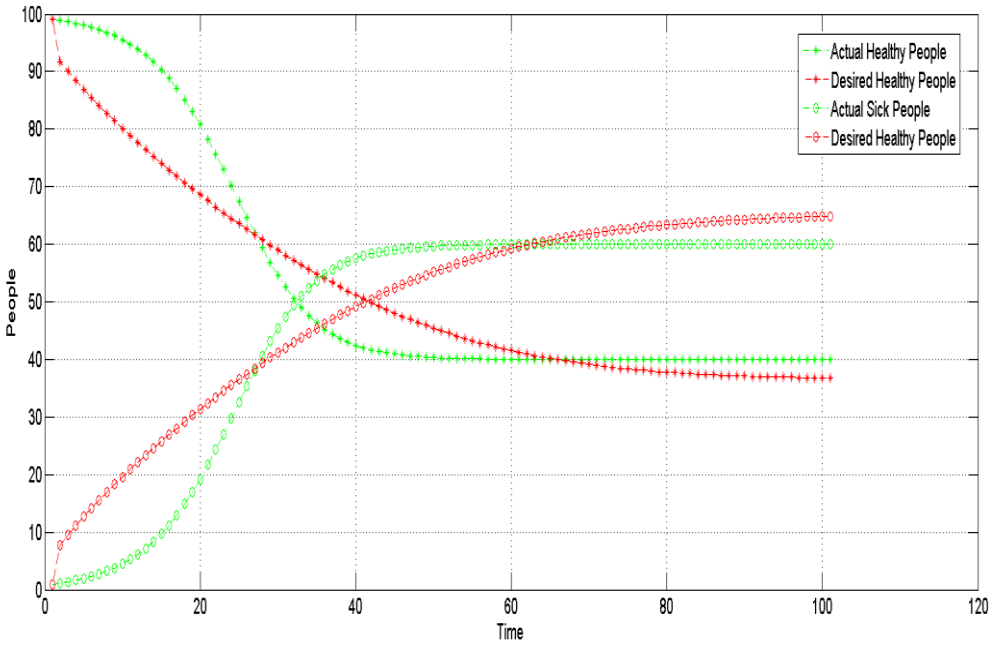

Figure 12: Desired healthy people and sick people compared with actual healthy and sick people

As shown in Figure 12, GP was also able to find close solution for both healthy people and sick people equations, as seen from Figure 10. The behaviour for the desired equations is not following the same actual behaviour at the early values, but after that it becomes very close. At the level of equations, we can see that the structures are also close together as the following:

Actual equations:

$$\left(\frac{\mathrm{SP}}{0.5} - \frac{5 * \mathrm{HP} * \mathrm{SP}}{\mathrm{SP} + \mathrm{HP}}\right)$$

$$\left(\frac{5 * \mathrm{HP} * \mathrm{SP}}{\mathrm{SP} + \mathrm{HP}} - \frac{\mathrm{SP}}{0.5}\right)$$

Best solution equations:

$$0.14 * \frac{SP^2}{HP} - \frac{HP}{SP} - 13.27$$

$$\frac{HP}{SP} - 0.14 * \frac{SP^2}{HP} + 17.045$$

5.2.2 Partial information conditions

In this experiment the reconstruction is accomplished by assuming that we have only the sick people observation and apply on it the reconstruction steps to produce other observations. The following steps describe how the reconstruction is accomplished in selecting the time lag and embedding dimension. Based on their values the reconstruction is accomplished on the sick people data to produce another observations that will be used as the actual observations. We apply this step to see if the GP will produce similar structure by applying it in the original observations.

5.2.2.1 Estimating time lag

Mutual information method is applied for estimating the time lag and the best value found is 2, Figure 13 shows the mutual information curve over different time lags.

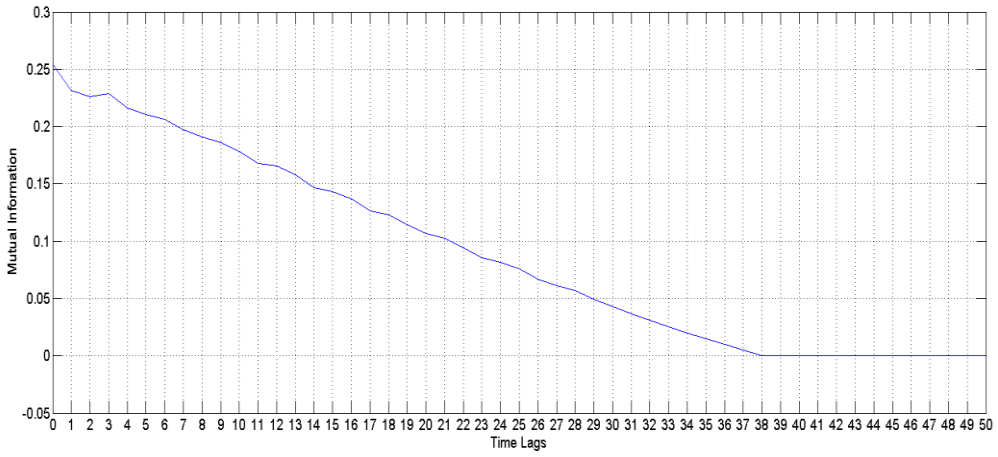

Figure 13: Mutual information over different time lags

5.2.2.2 Estimating embedding dimension

False nearest method is applied for estimating the embedding dimension and the best value found is 2 by using threshold value of 0.1, Figure 14 shows the false nearest curve over different embedding dimensions.

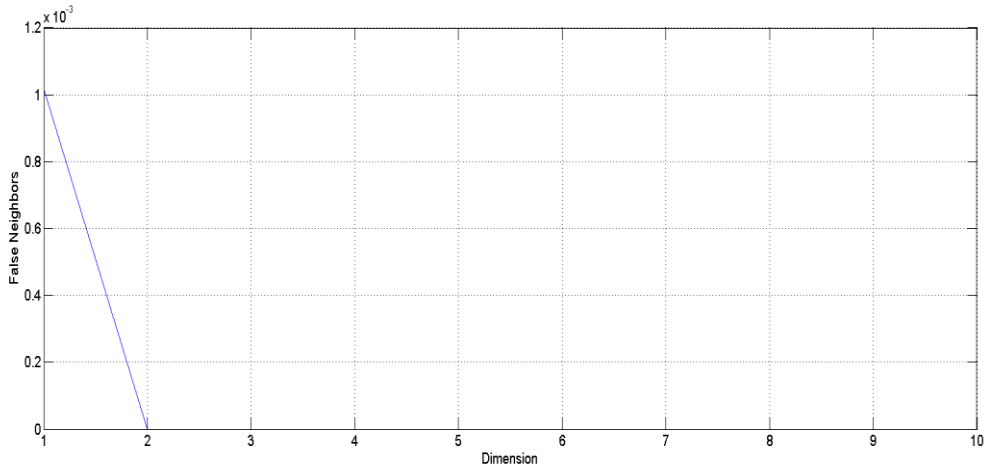

Figure 14: False neighbours over different dimensions

5.2.2.3 Reconstruction

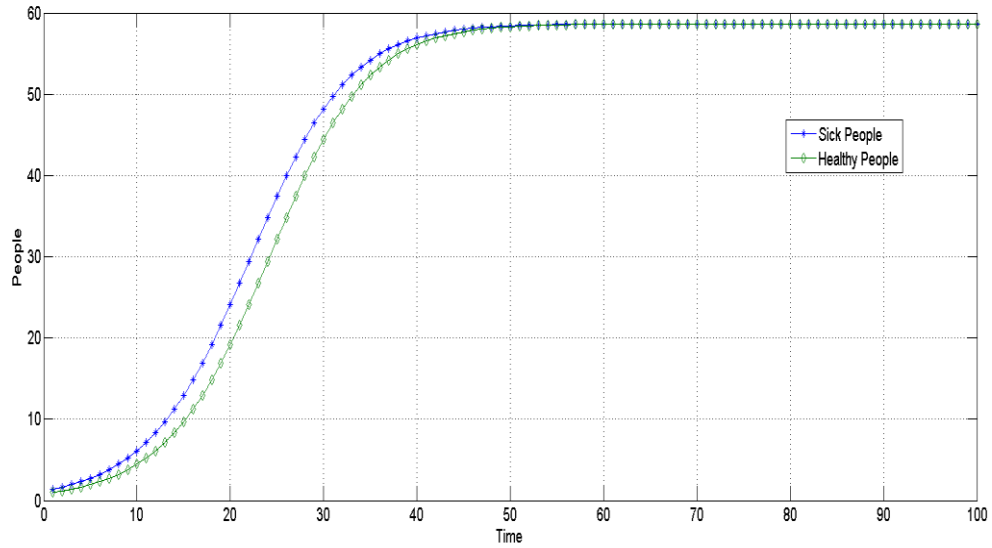

Figure 15: Reconstructed observations for sick and healthy people

Figure 15 shows the result of reconstruction step by using time delay and embedded dimension, we can see that the sick people observation is look similar to the original observation, but the healthy people reconstructed observation looks different from original healthy people observations, this may be due to that the reconstructed signals are produced mainly from the original observation and based on the estimated reconstructed parameters they can be similar to each other.

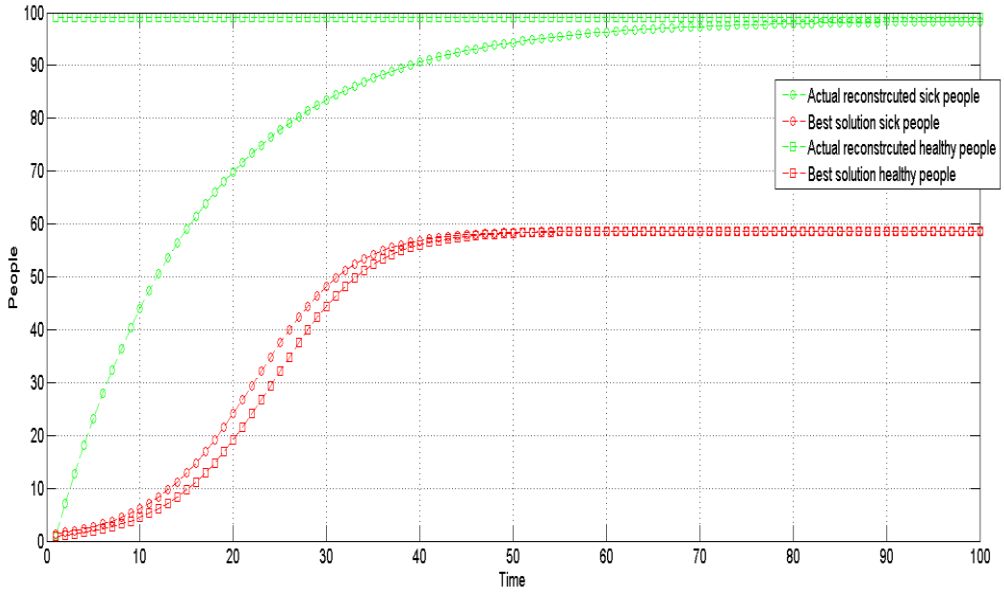

Figure 16: Best solution sick people and healthy people compared with actual reconstructed sick and healthy people

Figure 16 shows the best solution output compared with the reconstructed observations, it seems they follow the same behaviour but they are far away from each other. This also at the level of equations when compared with the actual model equations, but there are some similarities too as the following:

Actual equations:

$$\left(\frac{\mathrm{SP}}{0.5} - \frac{5 * \mathrm{HP} * \mathrm{SP}}{\mathrm{SP} + \mathrm{HP}}\right)$$

$$\left(\frac{5 * \mathrm{HP} * \mathrm{SP}}{\mathrm{SP} + \mathrm{HP}} - \frac{\mathrm{SP}}{0.5}\right)$$

Best solution equations:

$$\frac{HP}{exp^{\max((0.9-SP),\max(0.79,HP))}}$$

$$HP - SP - 0.5 - \frac{0.62}{HP}$$

5.3 Case 3

5.3.1 Full information conditions

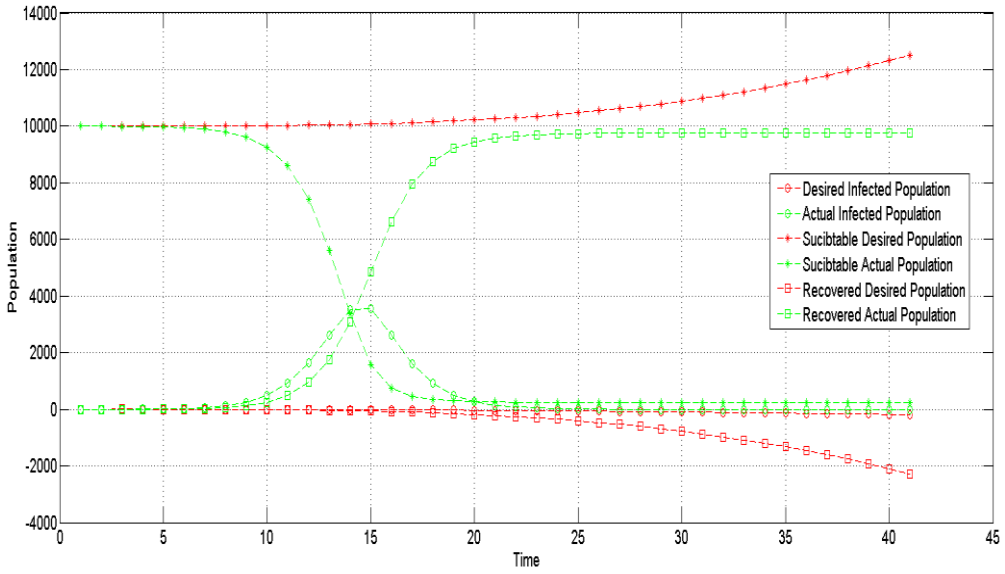

Figure 17: Desired infected, susceptible and recovered population compared with actual infected, susceptible and recovered population

As shown from Figure 17, the GP was not able to match the actual behaviour. In terms of the mathematical equations, we can also see that it simpler that the actual equations as the following:

Actual equations:

$$\left(\frac{1.5 * \text{SP} * \text{IP}}{(\text{IP} + \text{SP} + \text{RP})} - \frac{\text{IP}}{2}\right)$$

$$\left(-\frac{1.5 * SP * IP}{(\text{IP} + \text{SP} + \text{RP})}\right)$$

$$\left(\frac{IP_t}{2}\right)$$

Best solution equations:

$$0.57 - \frac{RP}{IP}$$

$$\frac{-IP}{0.98}$$

$$IP + \frac{RP}{IP}$$

5.3.2 Partial information conditions

In this experiment the reconstruction is accomplished by assuming that we have only the infected observation and apply on it the reconstruction steps to produce other observations, the following steps describes how the reconstruction is accomplished in selecting the time lag and embedding dimension. Based on their values the reconstruction is accomplished on the infected population data to produce three new values that is used as the actual observations that GP used to find the equations. We apply this step to see if the GP will produce similar structure to applying GP in the original observations.

5.3.2.1 Estimating time lag

Mutual information method is applied for estimating the time lag and the best value found is 8, Figure 18 shows the mutual information curve over different time lags.

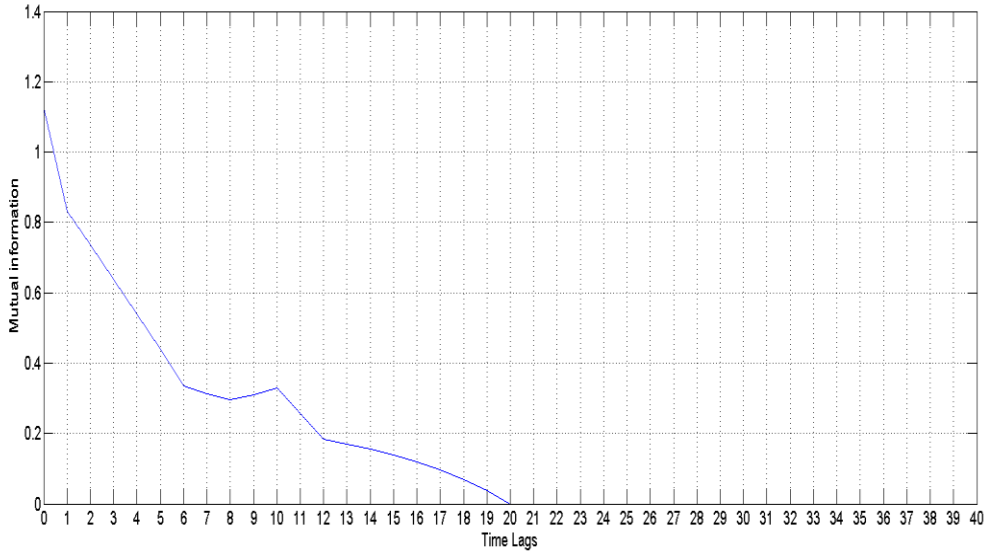

Figure 18: Mutual information over different time lags

5.3.2.2 Estimating embedding dimension

False nearest method is applied for estimating the embedding dimension and the best value found is 3 by using threshold value of 0.1, Figure 19 shows the false nearest curve over different embedding dimensions.

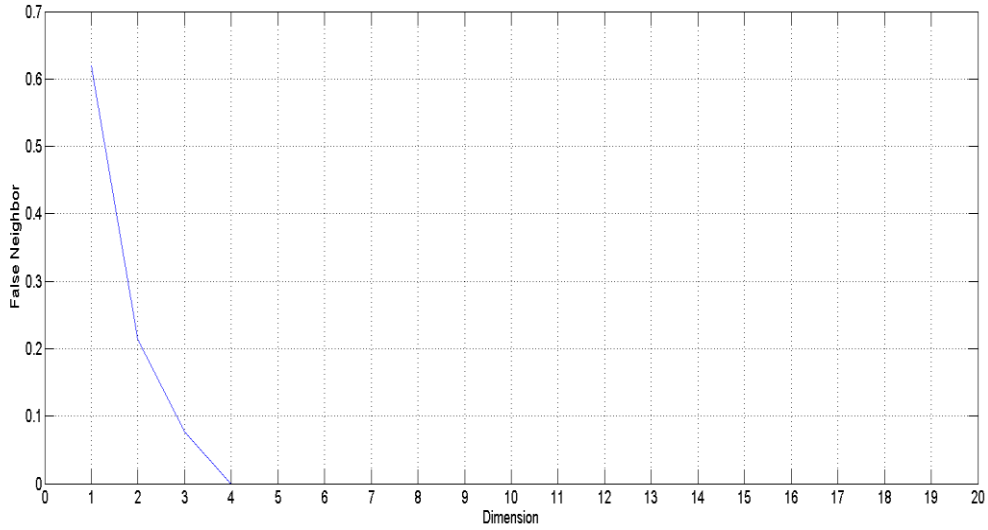

Figure 19: False nearest over different dimensions

5.3.2.3 Reconstruction

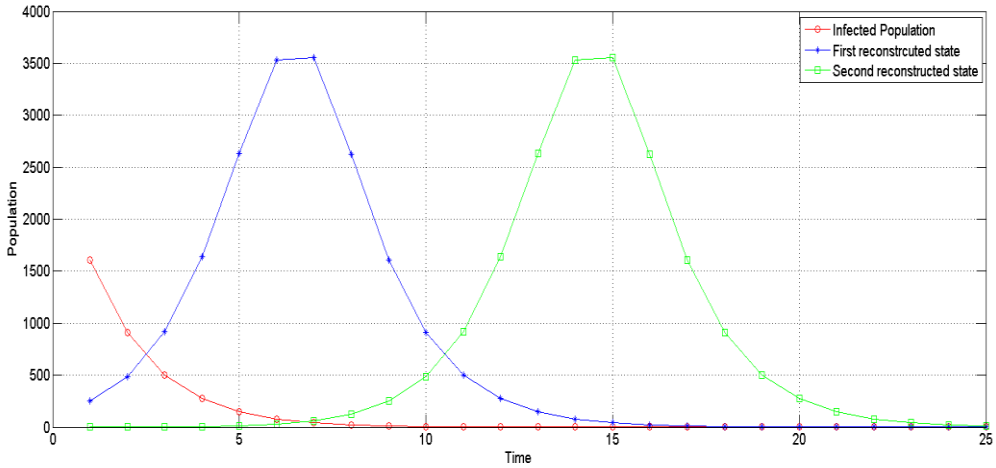

Figure 20: Reconstructed observations for infected, susceptible and recovered populations

Figure 20 shows the reconstructed observations from the infected population observation. When we compare it with the original observations we can see that the behaviour of the reconstructed two observations is much similar to the original infected population observations, but the other observations are different from the original observations of the model.

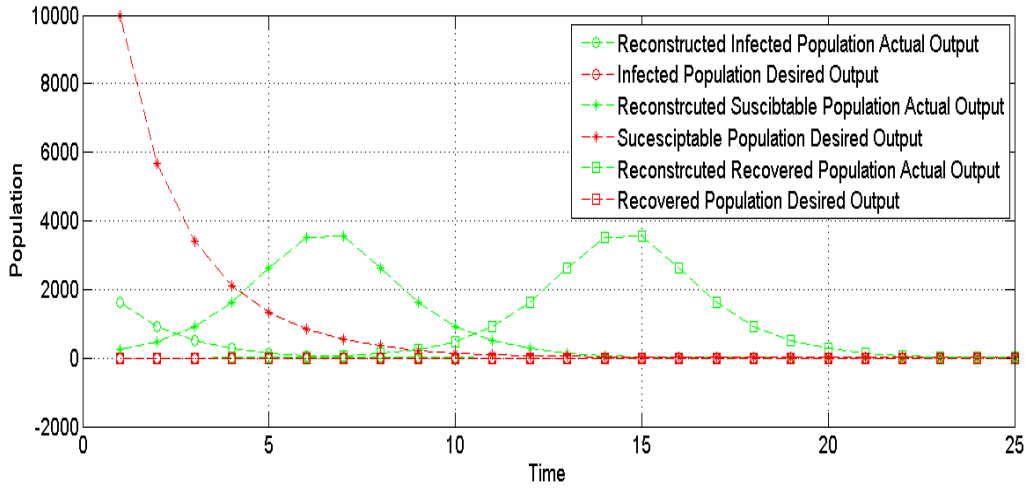

Figure 21: Desired infected, susceptible and recovered population compared with actual reconstructed infected, susceptible and recovered population

For the second case of applying the reconstruction step on only one given observation for the infected population, the GP was able to match one of the behaviours of the stocks as shown in Figure 21, but the other two behaviours was not captured properly. At the level of equations structure we can see that equations structure seems more complicated than the equations found in the case of no applied reconstruction as the following:

Actual equations:

$$\left(\frac{1.5 * \text{SP} * \text{IP}}{(\text{IP} + \text{SP} + \text{RP})} - \frac{\text{IP}}{2}\right)$$

$$\left(-\frac{1.5 * SP * IP}{(\text{IP} + \text{SP} + \text{RP})}\right)$$

$$\left(\frac{IP_t}{2}\right)$$

Best solution equations:

$$\sqrt[2]{0.72 * IP} - 0.72 * IP$$

$$IP + \frac{IP - 0.5 * SP}{IP}$$

$$\frac{IP^2}{IP - 0.5 * SP}$$

From all these results we can see that GP provided good results with one and two stocks models, but for the three stock models the behaviour generated starts to go far from the actual observations. In the case of applying the reconstruction step, the equations

structures were closer to the actual equations than in the case of not applying the reconstruction. It seems that applying the reconstruction to produce other observations may improve the results of finding the structure. This improvement may be due to the assumption that the observations we see from a system is the only generated and the most important ones, but there could be another observations behind the scenes play an important role in affecting the other observations. Applying the reconstructing process can be valuable in the cases such as, the inability to measure some of system observations or the lack of the availability of data.

We also can observe that the reconstruction step was able to estimate the correct dimension in all cases which provides a good indicator that we can rely on it to tell if there are other observations hidden. However, we can observe also that the reconstructed observations in case 2 and 3 are not much similar to the original observations they should be look like and this of course affecting the equations structures and make them different from structures obtained with full information case. Adding the reconstructed step is promising in handling the limited observations and can provide good guess of the how many other hidden observations are, but they can produce different behaviours that can affect the model learning process.

6 Conclusion and future work

This paper is a part of an ongoing research that aims to contribute in improving the efficiency and effectiveness of the model development lifecycle. We make use of machine learning techniques (i.e. GP and embedded reconstruction) to support learning about the model structure and parameters. Learning system dynamics model using computational models do not aim (and should not aim) to replace the modeller, but provide the technical support for the modeller to focus on the more important intellectual aspects of model building.

Our results show that employing GP was effective to find system dynamics mathematical equations with estimated parameters simultaneously and provide good results with first and second order models. The reconstruction of other observations from the provided ones can be valuable in discovering any other unmeasurable observations which can help in building system dynamics model in the case of limited observations. There is a need for other information beside the system observations to be used in model learning process to make sure the learned model is not just match the actual observation, but also keep the link between the structure and behaviour reserved.

From the results conducted we can say that the main strengths of the proposed methodology are represented in the ability of estimating the number of unmeasurable observations correctly, reconstructing them from original observation and the ability to learn both the equations and parameters simultaneously event with partial information provided. The main limitations are represented in the inability to reconstruct exact

model observations and the effect of higher orders models in the performance of the learning algorithm.

Our early findings indicate that leveraging the strengths of a toolbox of different computational methods is a promising direction to support learning about system dynamics model. The next step will be to apply the proposed methodology in large and more complex model, and investigating the use of other measures to compare the quality of the learning model solutions in addition to the given observations.

7 References

Abdollahiasl, A., et al. (2014). "A system dynamics model for national drug policy." DARU Journal of Pharmaceutical Sciences **22**(4).

Batista, G. E., et al. (2011). A Complexity-Invariant Distance Measure for Time Series. 2011 SIAM International Conference on Data Mining, Hilton Phoneix East/Mesa Mesa Arizona, USA.

Bianchi, C. and E. Bivona (2002). "Opportunities and pitfalls related to e-commerce strategies in small–medium firms: a system dynamics approach." System Dynamics Review **18**(3): 403-429.

Chen, J.-x. and B. Jeng (2009). AN EXPLORATION OF SYSTEM DYNAMICS IN BIOLOGICAL REVERSE ENGINEERING. Artificial Intelligence and Applications, Innsbruck, Austria

Chen, Y. T., et al. (2011). "A machine learning approach to policy optimization in system dynamics models." Systems Research and Behavioral Science **28**(4): 369-390.

de Lima, E. B., et al. (2010). Tuning Genetic Programming parameters with factorial designs. IEEE Congress on Evolutionary Computation.

Duggan, J. (2008). "Equation-based policy optimization for agent-oriented system dynamics models." System Dynamics Review **24**(1): 97-118.

Feldt, R. and P. Nordin (2000). Using factorial experiments to evaluate the effect of genetic programming parameters. Genetic programming, Springer**:** 271-282.

Ford, D. N. and J. D. Sterman (1998). "Dynamic modeling of product development processes." System Dynamics Review **14**(1): 31-68.

Fraser, A. M. and H. L. Swinney (1986). "Independent coordinates for strange attractors from mutual information." Physical review A **33**(2): 1134.

Garg, A., et al. (2014). "A computational intelligence-based genetic programming approach for the simulation of soil water retention curves." Transport in porous media **103**(3): 497-513.

Geyer-Schulz, A. (1996). "The MIT beer distribution game revisited: Genetic machine learning and managerial behavior in a dynamic decision making experiment." Genetic Algorithms and Soft Computing **8**: 658-682.

Graham, A. K. and C. A. Ariza (2003). "Dynamic, hard and strategic questions: using optimization to answer a marketing resource allocation question." System Dynamics Review **19**(1): 27-46.

Grossmann, B. (2002). Policy Optimisation in Dynamic Models with Genetic Algorithms. 20th International System Dynamics Conference Society Conference, Palermo, Italy, Citeseer.

Hegger, R., et al. (1999). "Practical implementation of nonlinear time series methods: The TISEAN package." Chaos: An Interdisciplinary Journal of Nonlinear Science **9**(2): 413-435.

Jeng, B., et al. (2006). "Applying data mining to learn system dynamics in a biological model." Expert Systems with Applications **30**(1): 50-58.

Kanninga, P. (2008). Simulation Model Development, The Devil is in the Detail!, TU Delft, Delft University of Technology.

Kennel, M. B., et al. (1992). "Determining embedding dimension for phase-space reconstruction using a geometrical construction." Physical review A **45**(6): 3403.

Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection, MIT press.

Langdon, W. B., et al. (2008). Genetic programming: An introduction and tutorial, with a survey of techniques and applications. Computational Intelligence: A Compendium, Springer**:** 927-1028.

Liu, H. (2012). Nature Inspired Computational Optimisation Methods for System Dynamics, National University of Ireland, Galway, Ireland.

Liu, H., et al. (2012). "Co-evolutionary analysis: a policy exploration method for system dynamics models." System Dynamics Review **28**(4): 361-369.

Meadows, D. H. (1989). "System dynamics meets the press." System Dynamics Review **5**(1): 69-80.

Medina-Borja, A. and K. S. Pasupathy (2007). Uncovering Complex Relationships in System Dynamics Modeling: Exploring the Use of CART, CHAID and SEM. 25th International Conference of the System Dynamics Society, Boston, Massachusetts, USA.

Önsel, N., et al. (2013). Evaluation of alternative dynamic behavior representations for automated model output classification and clustering. 31st International Conference of the System Dynamics Society, Cambridge, Massachusetts, USA.

Pawlas, K. and D. Zall (2012). "Analysis of forecasting methods and applications of system dynamics and genetic programming: Case studies on country throughput." Computer Science.

Poli, R., et al. (2008). A field guide to genetic programming, Lulu. com.

Pruyt, E., et al. (2014). From data-poor to data-rich: system dynamics in the era of big data. 32nd International Conference of the System Dynamics Society, Delft, The Netherlands, 20-24 July 2014; Authors version, The System Dynamics Society.

Robinson, S. (2004). Simulation: the practice of model development and use, John Wiley & Sons Chichester.

Sterman, J. D. (2000). Business dynamics: systems thinking and modeling for a complex world, Irwin/McGraw-Hill Boston.

Takahashi, Y. (2006). Stock Flow Diagram Making with Incomplete Information about Time Properties of Variables. 24th International Conference of the System Dynamics Society, Nijmegen, The Netherlands.

Takens, F. (1981). Detecting strange attractors in turbulence, Springer.

Willemain, T. R. (1994). "Insights on modeling from a dozen experts." Operations Research **42**(2): 213-222.

Winch, G. W. and D. J. Arthur (2002). "User-parameterised generic models: a solution to the conundrum of modelling access for SMEs?" System Dynamics Review **18**(3): 339-357.

Wolstenholme, E. F. (1999). "Qualitative vs quantitative modelling: the evolving balance." Journal of the Operational Research Society: 422-428.

Yücel, G. and Y. Barlas (2011). "Automated parameter specification in dynamic feedback models based on behavior pattern features." System Dynamics Review **27**(2): 195-215.