Article

A System Dynamics Modeling Support System Based on Computational Intelligence

Hassan Abdelbari * and Kamran Shafi

School of Engineering and Information Technology, University of New South Wales at Australian Defense Force Academy, Northcott Drive, Campbell, Canberra, ACT 2600, Australia; kamran.shafi@adfa.edu.au
* Correspondence: hassan.abdelbary@gmail.com

Received: 12 August 2019; Accepted: 4 September 2019; Published: 25 September 2019

Abstract: System dynamics (SD) is a complex systems modeling and simulation approach with wide ranging applications in various science and engineering disciplines. While subject matter experts lead most of the model building, recent advances have attempted to bring system dynamics closer to fast growing fields such as data sciences. This may prove promising for the development of novel support methods that augment human cognition and improve efficiencies in the model building process. A few different directions have been explored recently to support individual modeling stages, such as the generation of model structure, model calibration and policy optimization. However, an integrated approach that supports across the board modeling process is still missing. In this paper, a prototype integrated modeling support system is presented for the purpose of supporting the modelers at each stage of the process. The proposed support system facilitates data-driven inferring of causal loop diagrams (CLDs), stock-flow diagrams (SFDs), model equations and the estimation of model parameters using computational intelligence (CI) techniques. The ultimate goal of the proposed system is to support the construction of complex models, where the human power is not enough. With this goal in mind, we demonstrate the working and utility of the proposed support system. We have used two well-known synthetic reality case studies with small models from the system dynamics literature, in order to verify the support system performance. The experimental results showed the effectiveness of the proposed support system to infer close model structures to target models directly from system time-series observations. Future work will focus on improving the support system so that it can generate complex models on a large scale.

Keywords: modeling support system; system dynamics modeling; computational intelligence; genetic programming ensemble; simulated annealing

1. Introduction

System dynamics (SD) is a modeling and simulation approach with wide ranging applications in various disciplines such as management [1], constructions [2] and agriculture [3]. The need to improve efficiencies in the model building process and to deal with the explosion of data have already been acknowledged in the SD literature [4–6]. Several opportunities exist for recent advances in computational intelligence (CI) and machine learning techniques to be leveraged to support and improve the model building process. Although such techniques do not replace human cognition, they do alleviate some of the tedious tasks, such as data analysis, knowledge extraction, model calibration and model validation. This may lead to significant improvements in saving resources, dealing with large amounts of data and exploring large model spaces. The use of data driven approaches, specifically the CI methods, in SD modeling is not entirely new. In recent years, several studies have reported on the application of CI methods to SD modeling. Salient examples of such works include: the use of fuzzy logic (FL) for knowledge acquisition and representation [7] and identification of key system variables [8]; the application of echo state networks (ESNs) [9] and evolutionary echo state networks [10,11] for generating causal loop diagrams (CLDs); the use of a hybrid approach combining recurrent neural networks (RNNs) and genetic algorithms (GA) for generating stock and flow diagrams (SFDs) as well as for parameter estimation [12]; the application of genetic programming (GP) [13–15] and a hybrid of feed forward neural networks (FFNNs) and GA [16] for generating model equations and parameter estimation; the application of GA [17] , fuzzy multiple objective programming (FMOP) [18] and bootstrapping methods [19] for parameter estimation; the application of particle swarm optimization (PSO) [20], grammatical evolution (GE) [21], simulated annealing (SA) [22], decision trees [23] and differential games [24] for policy optimization; and the use of GA [25,26] and FL [27] for model validation and testing. A clear trend that can be observed from the above works is that the application of CI methods has been limited to specific stages of the model building process, with greater attention paid to model parameter estimation and policy optimization stages. Furthermore, the application of these methods to generate conceptual and formal models (often represented by CLDs and SDFs, respectively) or the underlying mathematical equations is rare.

This paper presents our progress towards developing an integrated CI-based, SD modeling support system to facilitate modelers through the model building stages. In particular, the learning engine of the support system builds upon several CI techniques, namely Cartesian genetic programming (CGP), GP ensemble learning, standard SA and fixed rule-based heuristics, to support the building of system CLDs, SFDs, equations and calibration of the model. A prototype of the proposed support system is provided as an easy to use executable tool. The tool provides a graphical user interface to feed data into the learning engine and to visually present the automatically inferred models. The minimum required knowledge that needs to be present before starting the automated modeling effort includes: the list of key variables, identification of stock variables, time series data capturing system observations for the stock variables, and the parameter settings of different CI methods.

Genetic programming (GP) is a population-based evolutionary algorithm, representing individual solutions as programs instead of a string of bits, as in a genetic algorithm (GA) [28], in which the evolving structures are represented as hierarchical trees. GP uses notions of functional (F) and terminal (T) sets to represent and evolve such structures, where all internal nodes are elements of F and all leaf nodes are elements of T. The generation of the initial population in a typical GP algorithm could be created in three different ways according to Koza [28]: full method, grow method and ramped half-and-half. Similar to other evolutionary algorithms, GP evolves the structures using selection, crossover, and mutation operators. The selection mechanisms most commonly used include tournament and roulette wheel selection. In crossover, random subtrees are selected from the parents and exchanged to generate the offspring. During mutation, each node is selected with a certain probability and the associated subtree is either replaced with a randomly generated one (subtree mutation) or the node type is simply changed (point mutation). Therefore, each solution has a fitness that measures the output error between target outputs and generated outputs. Standard GP suffers from several issues, the major one being bloating [29,30]. Bloating refers to the production of solutions with large amounts of unnecessary code that increases over the evolutionary process without providing any improvement [31]. Cartesian genetic programming (CGP) is a special, flexible and highly efficient type of GP algorithm that encodes a graphic representation of a computer program and uses only point mutation operator without applying the crossover operator [32]. It has several advantages over standard GP: it finds good solutions after only a few evaluations; it decodes solutions efficiently; it does not bloat; and it is easy to implement. Simulated annealing (SA) is a global optimization algorithm inspired from the cooling process of a material. During temperature reduction, the molecules of a material slow down and align to a crystallized form that represents the lowest energy state of the system [33]. Standard SA evolves one solution over time, unlike evolutionary algorithms that are population-based. The algorithm starts with an initial random solution, which is followed by a newly generated solution based on the initial one. The next step is the acceptance or rejection of the newly created solution. If the newly created solution shows an improvement in the objective function, it is accepted. If it shows an increase in the objective function, it is either rejected or accepted, with a probability based on the difference between objective values for current and newly created solutions. SA has the following advantages: able to deal with arbitrary cost functions, able to skip from local minimas, simple to implement, and can be applied to continuous and discrete problem domains.

We have demonstrated the ability of the modeling support system to generate useful models and equations by conducting experiments with two synthetic reality case studies taken from the SD literature. Our goal is to use CI methods in the SD field to support the modeling process with complex and large scale models. In this paper, however, we report on our progress using small models involving two and three stocks. In our opinion, the work presented here could be extended to improve the proposed support system and applied to complex models with a lot of data. To the best of our knowledge, this proposal is one of the first attempts to provide an SD modeling support system.

This paper is organized as follows: an overview of the proposed support system is followed by a description of how the different components work. The next section presents the experimental setup and describes the case studies used to demonstrate the performance of the support system. Lastly, we discuss the experimental results and limitations, draw conclusions and summarize future directions for this work.

2. Overview of SD Modeling Support System

Figure 1 shows the design of the proposed modeling support system using a conceptual block diagram. The support system consists of two main components: a learning engine, consisting of CI algorithms as the back-end of the system, and a graphical user interface component representing the front-end of the system. The graphical user interface component allows the modeler to interact with the tool, load the required input data, validate the format and correctness of the information provided and graphically present the generated models of the learning engine. The minimum information required by the system to generate the desired models includes the list of key variables, time series data capturing system observations for the stock variables, and the parameter settings of different CI methods. The learning engine is the core part of the support system and implements several CI algorithms to generate probable system CLDs, SFDs, model equations and parameters. The support system facilitates model building iteratively. In other words, the modeler can inspect the generated models and add, subtract or modify the input loaded into the data repository to see the effects of any changes. A user guide for the support system executable tool is provided in the supplementary material.

The support system can be invoked in two different modes: to generate the underlying equations and their parameters; or to generate CLDs, SFDs, equations and parameter estimation, as shown in Figure 2.

2.1. Inferring System Equations and Parameter Estimation

A hybrid genetic programming (GP) and simulated annealing (SA) ensemble method is applied to generate the stock variable differential equations and estimate their parameters, as shown in Figure 2. Given an expert-defined CLD, a GP ensemble is constructed and the variables' search space is decomposed to allow the inferring of each stock variable's differential equation independently. Each ensemble member consists of coupled Cartesian genetic programming (CGP) and SA algorithms to generate the equation structure and estimate its parameters, respectively. The tree structure for each GP ensemble member is created, based on the CLD. Four constraints are applied when creating any new individual, either at the initial population creation step, or after applying each mating operator, to ensure the feasibility of the evolved expressions: (1) the tree depth is restricted to not exceed the size of the terminal set by introducing a parameter, called the depth controller D_{contr}; (2) only the tree expression containing all the terms in the terminal set is accepted; (3) the duplication of symbols in the terminal set is disallowed; and (4) redundant expressions, such as $(X - X)$, (X/X), and $(X * 1)$, are disallowed. Algorithms 1 and 2, in the supplementary material, show CGP algorithm steps, applied for equation inferring, and standard SA algorithm steps, applied for parameter estimation, respectively.

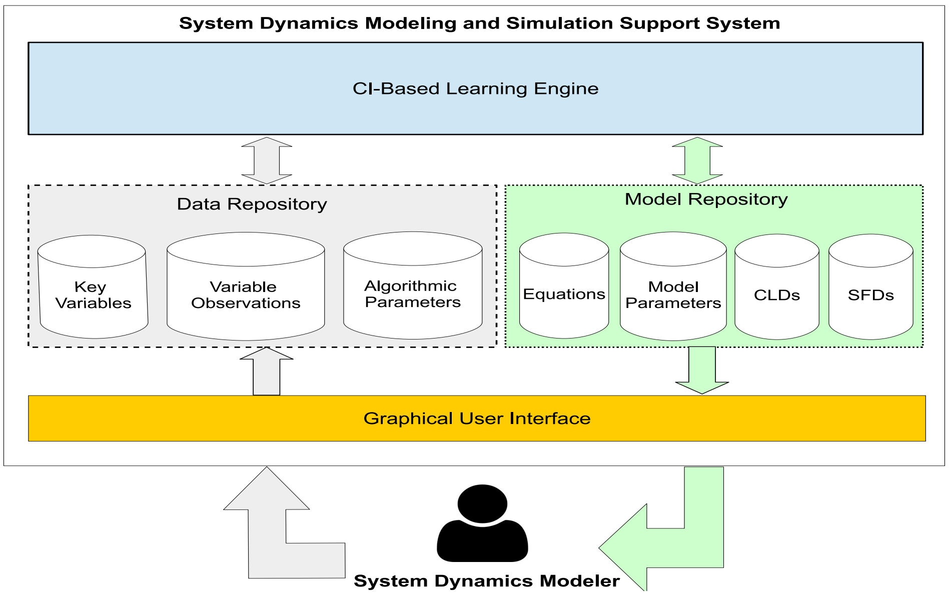


Figure 1. The proposed system dynamics modeling support system.

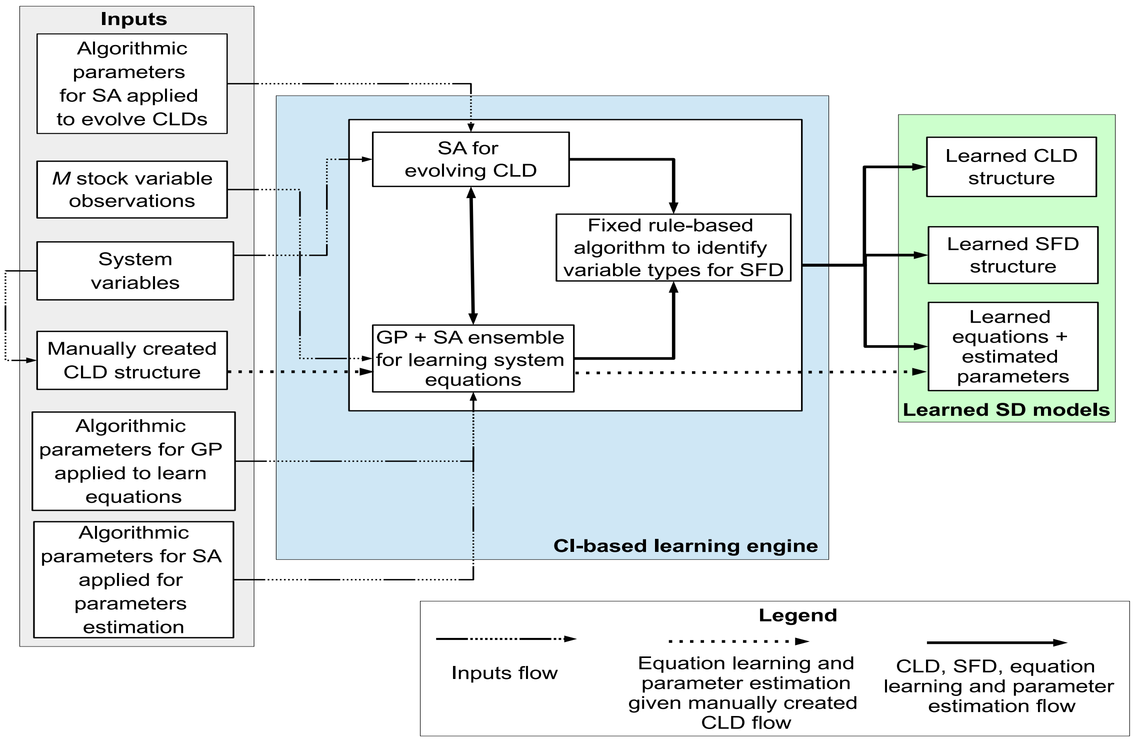

Inputs
Algorithmic parameters for SA applied to evolve CLDs
M stock variable observations
System variables
Manually created CLD structure
Algorithmic parameters for GP applied to learn equations
Algorithmic parameters for SA applied for parameters estimation
SA for evolving CLD
Fixed rule-based algorithm to identify variable types for SFD
GP + SA ensemble for learning system equations
CI-based learning engine
Learned CLD structure
Learned SFD structure
Learned equations + estimated parameters
Learned SD models
Legend
Inputs flow
Equation learning and parameter estimation given manually created CLD flow
CLD, SFD, equation learning and parameter estimation flow

Figure 2. The main components of the support system learning engine.

The equations generated for each ensemble member are combined. Following this, the outputs arising from simulating the generated equations are compared with the target outputs to calculate the output error (ϵ) by applying the complexity invariant distance (CID) measure [34]. Current distance measures, such as Euclidean distance and mean square errors, are introduced to measure the similarity between time series with invariance to various combinations of distortions [35], scaling [36], offset [37], phase [35], occlusions and uncertainty. However, the complexity invariance is still missing and not considered in all introduced measures. Time series data generated from complex systems obtain such complexity factor. Pairs of complex time series data, which could seem very similar, are considered further apart from each other under current standard distance measures. Complexity invariant distance measure uses information about complexity differences between two time series as a correction factor for Euclidean distance. CID is a Euclidean distance measure which when added to a correction factor measures the shape of the time series as follows:

$$\epsilon(X, \hat{X}) = \frac{\sum_{i=1}^{M} CID(X_i, \hat{X}_i)}{M}, \quad (1)$$

$$CID(X, \hat{X}) = ED(X, \hat{X}) \times CF(X, \hat{X}), \quad (2)$$

$$ED(X, \hat{X}) = \sqrt[2]{\sum_{i=1}^{L} (x_i, \hat{x}_i)^2}, \quad (3)$$

$$CF(X, \hat{X}) = \frac{max(CE(X), CE(\hat{X}))}{min(CE(X), CE(\hat{X}))}, \quad (4)$$

$$CE(X) = \sqrt[2]{\sum_{i=1}^{L-1} (x_i, x_{i+1})^2}, \quad (5)$$

where X is the target time series observations for the output variables, $\hat{X}$ the generated behavior from the inferred model's output variables, M is the number of output variables, L is the size of time series, ED is Euclidean distance, CF is the complexity correction factor and CE is the complexity estimate of a time series.

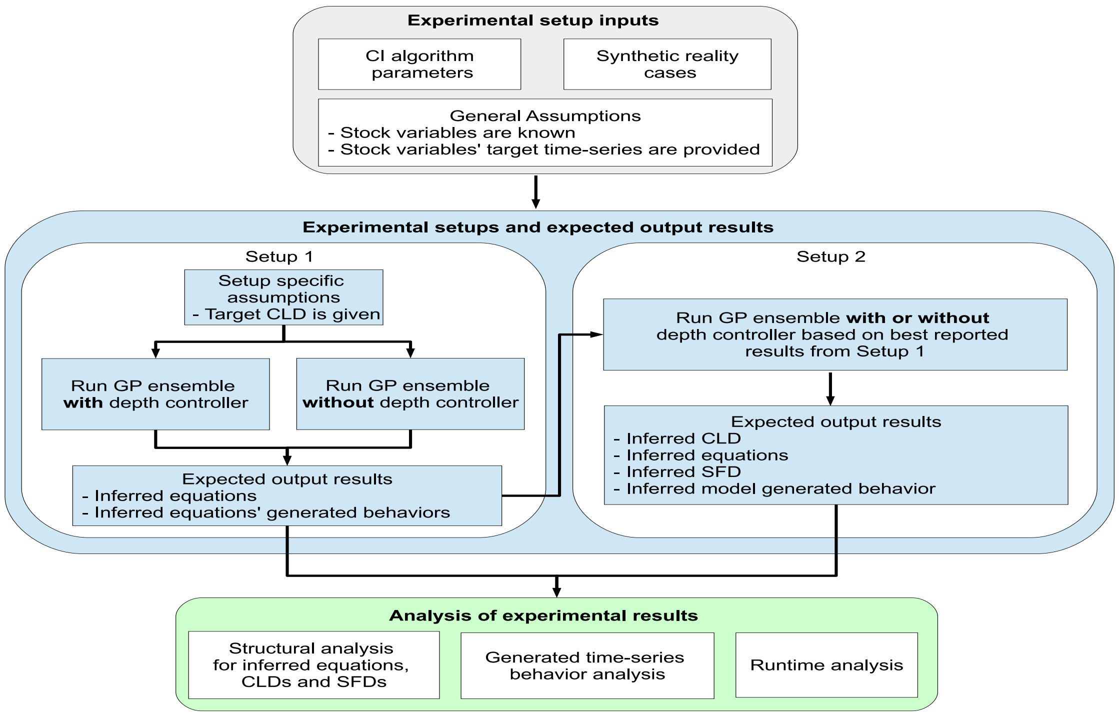
2.2. Inferring CLDs, SFDs, System Equations and Parameter Estimation

An integration between simulated annealing (SA) algorithm and the GP ensemble discussed in the previous subsection is applied to generate CLD, system equations and estimate the model parameters simultaneously. We represent the CLD using an adjacency matrix data structure [38]. During initialization, a CLD is created randomly from the set of variables provided by the modeler. We start with an empty adjacency matrix and for each ith row and jth column, a random link with probability ρ is created to describe a causal link from the ith variable to the jth variable, where ρ is a predefined CLD sparsity value. Before creating any link, we check the validity to create one between the two nominated variables based on the rules that describe how SFDs are created [39]. The quality of the CLD is determined by transforming the CLD into a simulation model by running the GP ensemble and calculating the CLD output error. A new CLD is then generated from the seed CLD using the SA variation operators by flipping each causal link between ith and jth variables with certain probability P_{flip}. Algorithm 3, in the supplementary material, shows the SA+GP ensemble integrated algorithm steps. Given the best generated CLD and equations with their estimated parameters, the SFD is generated by identifying the links among the variables and identifying the variable types to be either flows, auxiliaries or parameters, since we already know the stock variables. The variables that affect the stock variables directly are considered as flow variables. Based on the causal link polarity presented in the inferred CLD, this variable will be identified as either inflow for positive causal link or outflow for negative causal link. Finally, the remaining variables are considered as auxiliaries and the parameters estimated with the equations are included in the SFD.

3. Experimental Setup

This section provides the details of the experimental setup, which we used to validate the performance of the modeling support system in inferred SD models. Figure 3 shows the flow of the experimental setup starting with the inputs, setups and expected outputs from the experiments.

Experimental Setup inputs include the synthetic reality case studies; CI algorithm controlling parameters; and general assumptions regarding the setups. Two experimental setups are introduced, Setup 1 and Setup 2. We have assumed that the number of variables provided by the modeler was the complete set that efficiently described the system's behavior. Furthermore, we assumed that the types of stock variables are known and that time series observations will be available for all of them.



Experimental setup inputs
CI algorithm parameters
Synthetic reality cases
General Assumptions
- Stock variables are known
- Stock variables' target time-series are provided
Experimental setups and expected output results
Setup 1
Setup specific assumptions
- Target CLD is given
Run GP ensemble with depth controller
Run GP ensemble without depth controller
Expected output results
- Inferred equations
- Inferred equations' generated behaviors
Setup 2
Run GP ensemble with or without depth controller based on best reported results from Setup 1
Expected output results
- Inferred CLD
- Inferred equations
- Inferred SFD
- Inferred model generated behavior
Analysis of experimental results
Structural analysis for inferred equations, CLDs and SFDs
Generated time-series behavior analysis
Runtime analysis

Figure 3. Experimental setup.

3.1. Setups

3.1.1. Setup 1

The goal of the first set of experiments was to establish a baseline whether the genetic programming (GP) ensemble can generate similar equations with correctly estimated parameters to the target equations given the target CLD. In this setup, we ran the experiments to generate the equations with and without the depth controller parameter D_{contr}. This was done to determine whether application of the depth controller was helpful for generating similar equations.

The GP algorithm for each ensemble member performed 25 independent runs with different random seeds. We recorded the best inferred solution at each generation, based on the minimum fitness value. The best inferred solution was recorded over all generations and selected for the run. Finally, the solution with the minimum fitness value was selected from these 25 solutions to be the best of the best, over all runs. This process was run M times, where M is the case stock variables number used to create the ensemble members. This provided us with a total of $25 \times M$ runs. These runs were conducted twice, once by applying the depth controller parameter D_{contr} and then without applying it. For Case 1 (two stock variables) and Case 2 (three stock variables), the total number of runs is 100 ($2 \times 2 \times 25$) and 150 ($2 \times 3 \times 25$), respectively.

3.1.2. Setup 2

In this setup, we ran the experiments to simultaneously generate both the CLD and the equations with estimated parameters. Using the best inferred models, we were able to generate the SFD. The aim of this setup was to test the ability of the method under development to generate models similar to those of the system's target. Based on the results from Setup 1, the experiments were run with or without depth controller.

The simulated annealing (SA) algorithm, integrated with the GP ensemble and depth controller setting, made 10 independent runs with random seeds. In order to select the best overall, the best inferred solution was recorded at each SA iteration, based on the minimum fitness value. Then, the best inferred solution of all iterations was selected to be the best one for this run. Finally, the solution with the minimum fitness value was selected to be the best of the best, over all runs.

3.2. Case Studies

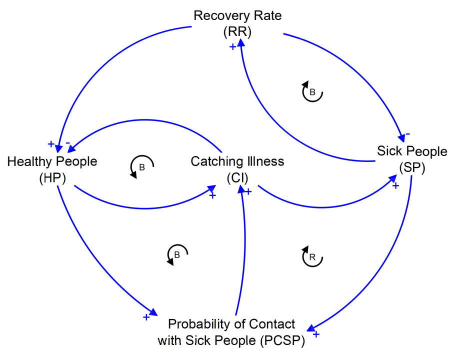
Two synthetic reality case studies were used for validation in each of the experimental settings. These cases were selected to provide variety in the number of stocks, as well as complexity in the CLDs and SFDs, equations, and the behavioral patterns. The target outputs for the two cases were generated through building and simulating the standard SFDs. For each case, we implemented our own simulation model by solving the model equations using the Euler method. Python source code for the implemented simulation models of the two cases are provided in the supplementary material. The delta time (DT) values used for simulating Case 1 and Case 2 are 0.0625 and 0.58, respectively. These same values are used to simulate the inferred equations by applying GP ensemble method.

Case 1 is a simple epidemic model that shows how healthy people are converted to sick people [40]. The model has two stocks: healthy people (HP), and sick people (SP); two flows: catching illness (CI) and recovering rate (RR); one auxiliary variable: probability of contact with sick people $(PCSP)$; and three exogenous inputs (model parameters): population interactions (PI), probability of catching illness (PCI) and duration of illness (DI), as shown in the CLD (Figure 4a) and SFD (Figure 4c). The simulated stocks show S-shape growth behaviors (Figure 4e).

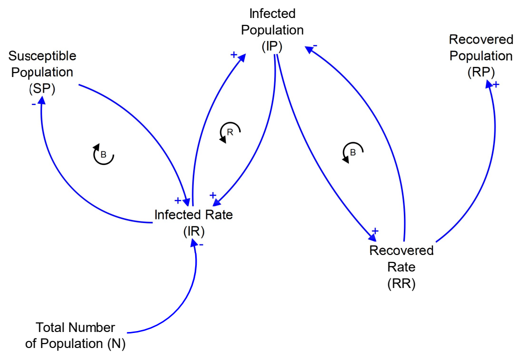
Case 2 is a more complex epidemic model that shows how a population goes through different states, from susceptible to infected and finally to recovery [41]. The model has three stocks: susceptible population (SP), infected population (IP), and recovered population (RP); two flows: infection rate (IR) and recovery rate (RR); and four exogenous inputs (model parameters): total number of population (N), infectivity (i), contacts (c) and duration (d), as shown in the CLD (Figure 4b) and SFD (Figure 4d). The simulated stocks show different behaviors that include S-shape growth behavior and S-shape growth with decline behavior (Figure 4f).

3.3. CI Algorithm Parameters

For the CGP algorithm used by each GP ensemble member, defining the functional and terminal sets and the parameter ranges are case-dependent given our pre-defined assumptions of the target models. The functional set $\{+, -, *, \%\}$ is used for both Case 1 and Case 2. The parameter ranges, identified for the SA algorithm, applied to estimate the parameters of the generated equations are $[1, 6]$ and $[0, 2]$ for Case 1 and Case 2, respectively. For each ensemble member, the terminal sets are defined for each case given the CLD. The remaining parameters for GP algorithm are defined given the recommendations in [28], where the population size P_{size} is 50, and the maximum number of generations G is 100. The grow method is applied to generate the initial population with varying depth D ranging from 1 to 3, and the usage of the depth controller D_{contr}. Tournament selection mechanism is applied with size $Tour_{size}$ equals to 3, and a single point mutation is used with probability P_m of 0.05.

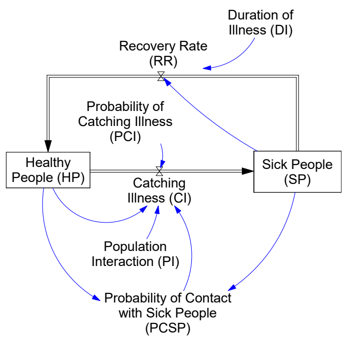


(**a**) Case 1 target CLD.



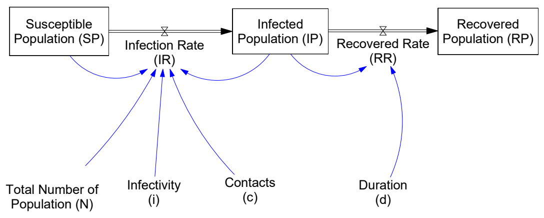
Susceptible Population (SP)
Infected Population (IP)
Recovered Population (RP)
Infected Rate (IR)
Recovered Rate (RR)
Total Number of Population (N)
B
R

(**b**) Case 2 target CLD.

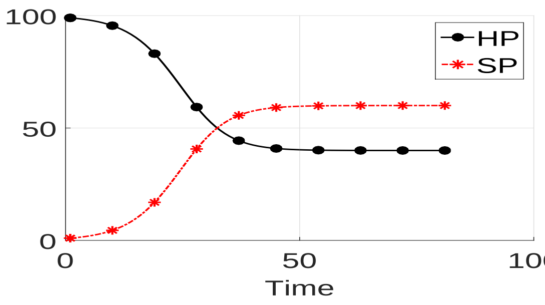


Duration of Illness (DI)
Recovery Rate (RR)
Probability of Catching Illness (PCI)
Healthy People (HP)
Catching Illness (CI)
Sick People (SP)
Population Interaction (PI)
Probability of Contact with Sick People (PCSP)

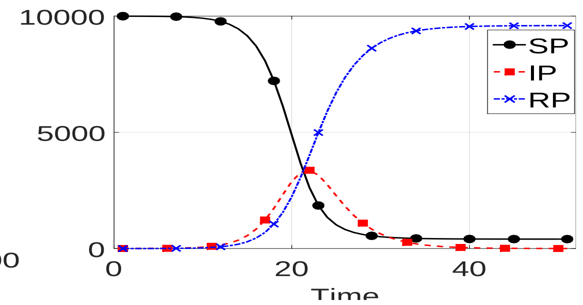
(**c**) Case 1 target SFD.



(**d**) Case 2 target SFD.



(**e**) Case 1 target behavior.



(**f**) Case 2 target behavior.

Figure 4. Case studies system dynamics models and behaviors.

For SA algorithms applied to estimate inferred equation parameters and to generate the CLD, we used an initial temperature T_0 of 10,000 and cooling constant α of 0.95. For the SA algorithm that estimates the inferred equation parameters, the perturbation constant p_{const} is equal to 2.5 and the maximum number of iterations and sub iterations is equal to 10 and 3, respectively. For the SA algorithm that is applied to learn the CLD in combination with GP ensemble, the links flipping probability P_{flip} is equal to 0.05; the maximum number of iterations and sub iterations is equal to 100 and 15 for Case 1 and Case 2, respectively; and the CLD's sparsity ρ is equal to 0.0981. This sparsity value is estimated by calculating the average sparsity, which is the total number of links divided by all possible links, for target CLDs for both cases. The parameters used for the SA algorithms are based on values recommended in the academic literature [33].

4. Results and Analysis

4.1. Setup 1

Setup 1 results for Case 1 and Case 2 are shown in Tables 1 and 2, respectively. Each equation set shows the target equations, followed by equations generated by the GP ensemble in the presence and absence of the depth controller parameter D_{contr}. The observation was made that each stock variable's generated equation contained the correct variables. This was because the knowledge of causalities among the variables was known from the predefined CLD. This knowledge defines the terminal set for each GP ensemble member and determines how the generated equations will look. The comparison between target and generated behaviors from Case 1 inferred equations (Table 1) and Case 2 inferred equations (Table 2), with and without applying the depth controller, is shown in Figures 5 and 6, respectively.

4.1.1. Case 1

For Case 1 with and without the depth controller, an obvious difference between the target equations and generated ones resides in the values of model parameters in all generated equations. Additional parameter values were added to the SP stock variable's generated equations in the presence and absence of D_{contr}. In addition, additional parameter values were added to the HP stock variable's generated equation only in the absence of D_{contr}. For the generated equation structures, the GP ensemble is able to generate equations for both HP and SP stock variables similar to the target equations in the presence and absence of the depth controller parameter. However, the denominator term $(HP + SP)$ is missing in the generated equations, as shown in Table 1. Since this term is the sum of both populations (total number of population), it could be considered a normalization term. The inferred equations with depth controller generate behaviors more similar to those of the targets than in the absence of depth controller, as shown in Figure 5.

Table 1. Comparison of Case 1 target system equations with inferred equations using genetic programming ensemble applied with and without depth controller parameter D_{cont}—Setup 1.

Target Equations	**Inferred with $D_{cont} = 1$**	**Inferred with $D_{cont} = 0$**
$\frac{dHP}{dt} = (2 \times SP) - (\frac{5 \times HP \times SP}{HP+SP})$	$\frac{dHP}{dt} = (1.99 \times SP) - (0.05 \times HP \times SP)$	$\frac{dHP}{dt} = (1.95 \times SP) - (0.05 \times HP \times SP - 21.74)$
$\frac{dSP}{dt} = (\frac{5 \times HP \times SP}{HP+SP}) - (2 \times SP)$	$\frac{dSP}{dt} = (0.04 \times HP \times SP + 1.58) - (1.77 \times SP)$	$\frac{dSP}{dt} = (0.05 \times HP \times SP + 0.41) - (1.9 \times 8SP)$

A possible explanation for the absence of this term from the generated equations using depth controller is that the depth controller restricts the GP generated tree to not exceed the terminal set size. This encourages the GP to converge toward solutions with small and balanced trees. This is illustrated by the term $(0.05 \times HP \times SP)$ in the HP generated equation where its tree size is one and the first multiplication operator is the tree root. In the SP generated equation, the tree size is two and the second multiplication operator is the tree root, as shown by $(0.04 \times HP \times SP + 1.58)$ term. For the generated equations, without the depth controller, the absence of this term causes the generation of large and more unbalanced GP trees, as shown by the term $(0.05 \times HP \times SP - 21.74)$ in the HP generated equation. Its tree size is three and the first multiplication operator is the tree root, similarly for the term $(0.05 \times HP \times SP \times 0.41)$ whose tree size is three and the plus operator is the tree root. In terms of model parameters and equation structures, there is no obvious advantage to apply the depth controller over not applying it. However, we consider it best to apply the depth controller to the generated equations because the resulting behavior is closer to the target behavior.

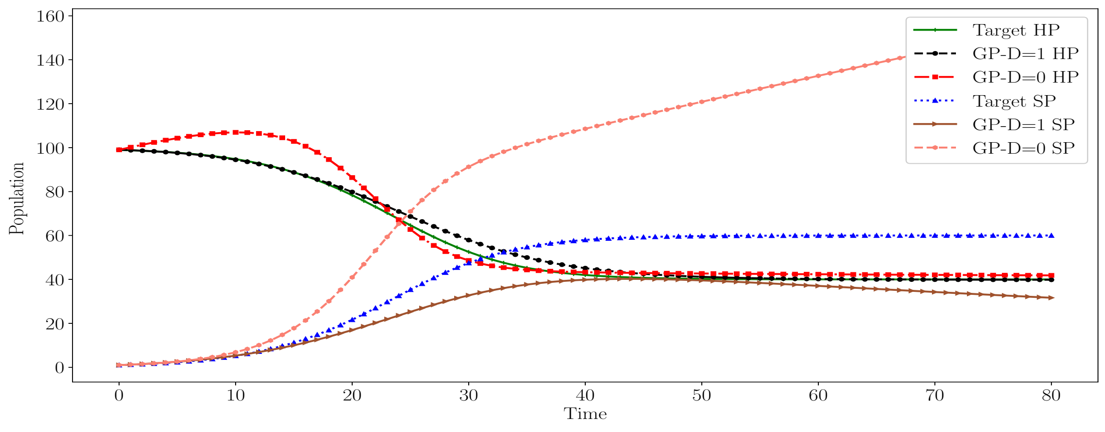


Figure 5. Illustration of Case 1 target behaviors, healthy people (HP) and sick people (SP), and the generated behaviors from the corresponding inferred models for Setup 1, with and without applying genetic programming depth controller.

4.1.2. Case 2

For Case 2, the GP ensemble is able to generate the exact target equations, including the correct values for model parameters, with and without the depth controller, as shown in Equation (2). A possible explanation for the ability of GP to generate these exact similar structures is that the target equation has maximum depth of two with smaller algebraic operators compared to the operators in Case 1's equation. GP, with and without the depth controller, results in small and balanced trees similar to the target equations. The generated behaviors from inferred equations exactly match the target behaviors, as shown in Figure 6. This is expected since the inferred equations exactly match the structure of the target equations.

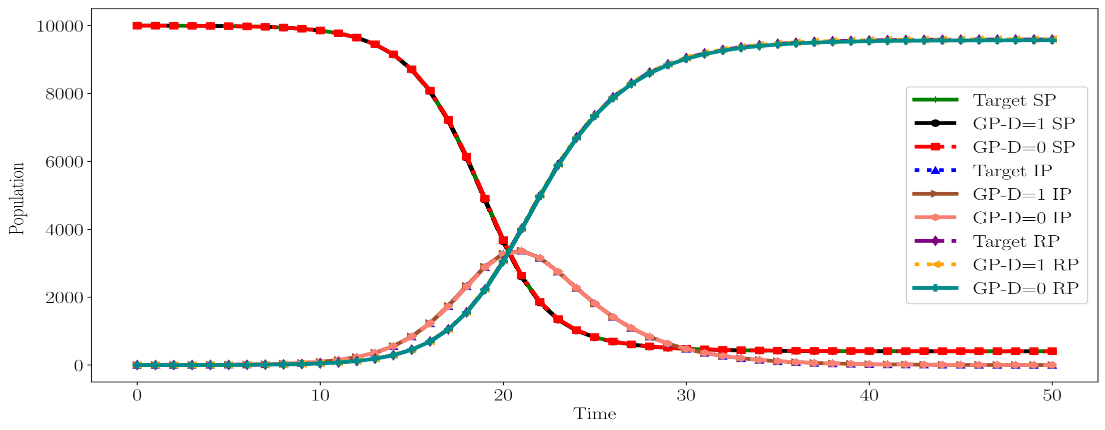


Figure 6. Illustration of Case 2 target outputs, susceptible population (SP), infected population (IP) and recovery population (RP), and the two generated outputs from the corresponding inferred models for Setup 1, with and without applying genetic programming depth controller.

Table 2. Comparison of Case 2 target system equations with inferred equations using genetic programming ensemble applied with and without depth controller parameter D_{cont}—Setup 1.

Target Equations	**Inferred with $D_{cont} = 1$**	**Inferred with $D_{cont} = 0$**
$\frac{dSP}{dt} = -\frac{1.5 \times SP \times IP}{N}$	$\frac{dSP}{dt} = -\frac{1.5 \times SP \times IP}{N}$	$\frac{dSP}{dt} = -\frac{1.5 \times SP \times IP}{N}$
$\frac{dIP}{dt} = \frac{1.5 \times SP \times IP}{N} - 0.5 \times IP$	$\frac{dIP}{dt} = \frac{1.501 \times SP \times IP}{N} - 0.5 \times IP$	$\frac{dIP}{dt} = \frac{1.5 \times SP \times IP}{N} - 0.501 \times IP$
$\frac{dRP}{dt} = 0.5 \times IP$	$\frac{dRP}{dt} = 0.5 \times IP$	$\frac{dRP}{dt} = 0.5 \times IP$

4.2. Setup 2

For each case in Setup 2, we compared the best generated CLD, SFD and equation structures with the target structures. The comparison between target and generated behaviors from Case 1 inferred equations (Table 3), with and without applying the depth controller, is shown in Figure 7. The best generated CLD for Case 1 was compared to the target structure by illustrating the correctly predicted, additional and missing links, as shown in Figure 8. The best generated SFD for Case 1 was compared to target one, as shown in Figure 9. The comparison between target and generated behaviors from Case 2 inferred equations (Table 4), with and without applying the depth controller, is shown in Figure 10. The best generated CLD for Case 2 was compared to the target structure by illustrating the correctly predicted, additional and missing links, as shown in Figure 11. The best generated SFD for Case 2 was compared to target one, as shown in Figure 12. In contrast to Setup 1, in Setup 2, the defined terminal sets for each GP ensemble member depend upon the evolved CLD links by the SA algorithm. We generated SFD by applying predefined rules to mark variable types as either stock, flow, auxiliary or model parameters.

4.2.1. Case 1

An obvious difference between the target equations and generated equations, in Case 1, lies in the values of the model parameters for the generated equations. Additional parameter value is added to the second term of the HP stock variable's generated equation. The inferred CLD has six correct links (Figure 8b) out of nine, three missing (Figure 8d) and four additional (Figure 8c). It can be seen from the values of different generated links in the CLD that the SA algorithm is able to predict most of the correct links. This is because it is a global optimization algorithm. The constructed SFD for this case (Figure 9b) is quite similar to the target one with some additional links and flow variables. This is because the SFD is based on the generated CLD and equations. Evolving the CLD by flipping randomly chosen links to either on or off affects the defined terminal sets for the GP ensemble that also affects the evaluation of how accurate the CLD is. The accuracy of CLD is the output error between the behavior generated from inferred equations and the target ones. Given the best generated CLD, the terminal sets defined for the GP ensemble that generates the HP stock equation will be $\{SP, HP\}$ for both equation terms. The terminal sets defined for the GP ensemble that generates the SP stock equation will be $\{SP, HP\}$ for first equation term and SP for second equation term. The presence of HP symbol in the first term of HP stock variable's generated equation could be explained by the defined terminal set $\{SP, HP\}$ for both terms in HP stock generated equation and the restriction for each GP ensemble member to include all symbols defined in the terminal set. This provide us with correct symbols in the second term of HP stock variable's generated equation. Given the correct defined terminal sets for SP stock variable's generated equation terms, the generated equation is quite similar to the target one. However, we made the same observation with Setup 1: in the absence of the denominator term $(HP + SP)$, for both stocks, HP and SP generated equations. This is the effect of applying the depth controller to restrict the tree size to the size of the terminal sets. In the absence of this term, the equation structure still valid since it could be considered as a normalization term. The generated behaviors from inferred equations exactly match the target. This is despite the addition of HP symbol in the first term of stock variable HP inferred equation and the minus operator between the symbols in the second term. This observation was not expected since the inferred equations in Setup 1 looks more similar to the target equations than the inferred equations in Setup 2.

Table 3. Comparison of Case 1 target system equations with inferred equations using genetic programming ensemble applied with depth controller parameter D_{cont}—Setup 2.

Target Equations	Inferred with $D_{cont} = 1$
$\frac{dHP}{dt} = (2 \times SP) - (\frac{5 \times HP \times SP}{HP+SP})$	$\frac{dHP}{dt} = (-0.05 \times SP \times HP) - (HP - SP - 99.968)$
$\frac{dSP}{dt} = (\frac{5 \times HP \times SP}{HP+SP}) - (2 \times SP)$	$\frac{dSP}{dt} = -(0.05 \times HP \times SP) + (2.001 \times SP)$

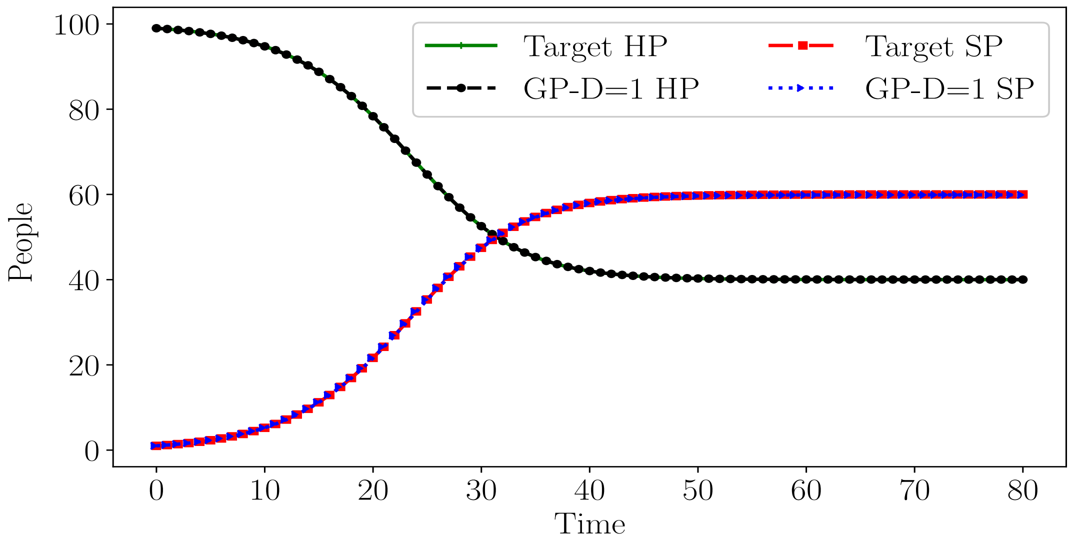


Figure 7. Illustration of Case 1 target outputs, healthy people (HP) and sick people (SP), and the two generated outputs from the corresponding inferred models for Setup 2 with genetic programming depth controller.

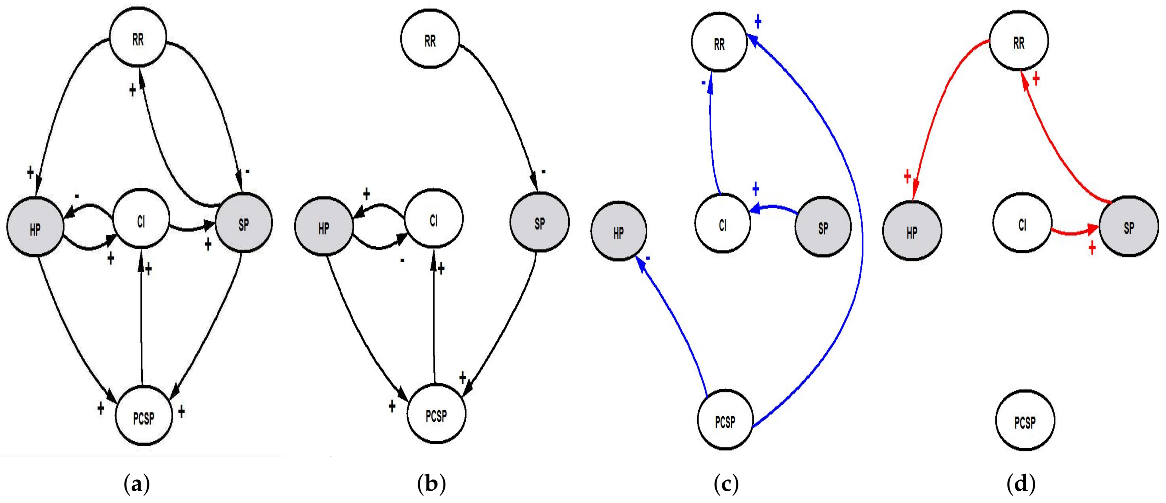
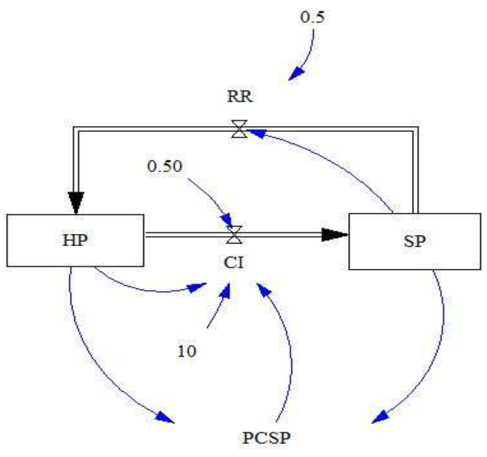
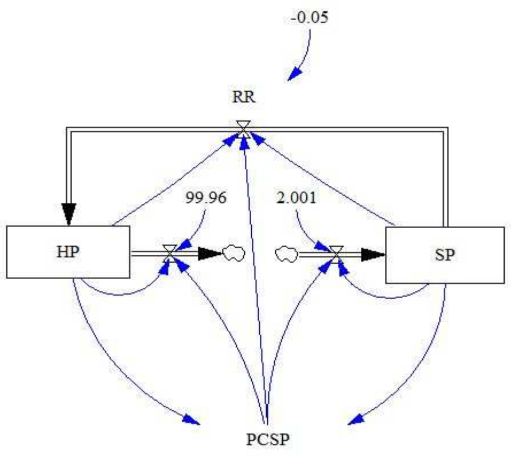


Figure 8. Case 1 inferred CLD by showing the correctly predicted links (**b**), additional links (**c**) and missing links (**d**) compared to the target CLD (**a**)—Setup 2.



(**a**) Target SFD.



(**b**) Inferred SFD.

Figure 9. Case 1 inferred SFD vs. target one—Setup 2.

4.2.2. Case 2

In Case 2, the differences between the target equations and generated equations reside in the values of model parameters. A possible explanation for these differences and additional parameter values is the presence of the additional symbols in these generated equations, specifically in the RP stock generated equation. The GP algorithm will have more numeric values if there are many symbolic terms in the tree. This is because the GP tree is constructed in a random manner from the terminal set and crossover and mutation operators. The inferred CLD has five correct links out of eight (Figure 11b), three missing (Figure 11d) and four additional (Figure 11c). Similar to Case 1, the SA algorithm is able to predict most of the correct links, causing the generated equations to be more similar to target equations. Since generating the SFD is based on the generated CLD and equations, the constructed SFD for this case (Figure 12b) closely matches the target one with two additional links. Given the best generated CLD, GP ensemble members that generated all stock variables' equations were using $\{SP, IP, N\}$ as their terminal set. The correctly defined terminal set for the GP ensemble that generates the SP stock equation enables generation of the same structure as the target equation—similarly for the correctly defined terminal set for the IP stock generated equation, which also results in the same structure as the target equation. However, the incorrect defined terminal set with additional symbols SP and N for the both IP and RP stock generated equations causes the presence of these symbols and generates different structures than target equations. Even with applying the depth controller to restrict the tree depth size, the presence of these symbols in the terminal sets forces the GP ensemble members to reject any tree structure lacking these symbols. The generated behaviors from inferred equations exactly match the target behaviors. This was not expected since the inferred equations have additional terms added to IP and RP stock variables' inferred equations.

4.3. Runtime Analysis

We used Java version 1.8.0_91 to develop the algorithms. The experiments were run on an Intel Core i7 CPU 3.4 GHz, with 16 GB of RAM and Windows 7 (64 bit) machine (Dell, Canberra, Australia). The execution time for one run of the GP ensemble for equation learning and parameter estimation (Setup 1) and SA + GP ensemble for inferring CLD (Setup 2) was recorded for each case, as shown in Table 5.

For Setup 1, cases 1 and 2 execution times are around 6.2 and 6.9 min, respectively. Cases 1 and 2 have five and six total variables, respectively, and two and three stock variables, respectively. Since the GP ensemble is constructed based on the number of stock variables, we believe that number of stock variables effects the execution time. In addition, the complexity of relationships among variables also has an effect on the execution time. We think this an acceptable time to learn such models, given the ability of the support system to navigate through the modeling space by producing a large number of models. In order to judge if the execution time of the GP ensemble is reliable, the time taken by a human modeler to come up with these models should be recorded.

Table 4. Comparison of Case 1 target system equations with inferred equations using genetic programming ensemble applied with depth controller parameter D_{cont}—Setup 2.

Target Equations	**Inferred with $D_{cont} = 1$**
$\frac{dSP}{dt} = -\frac{1.5 \times SP \times IP}{N}$	$\frac{dSP}{dt} = -\frac{1.5 \times SP \times IP}{N}$
$\frac{dIP}{dt} = \frac{1.5 \times SP \times IP}{N} - 0.5 \times IP$	$\frac{dIP}{dt} = \frac{SP \times IP}{N} - 0.00005 \times IP \times (N - SP)$
$\frac{dRP}{dt} = 0.5 \times IP$	$\frac{dRP}{dt} = \frac{83.96 \times IP \times SP + N}{167.91 \times SP}$

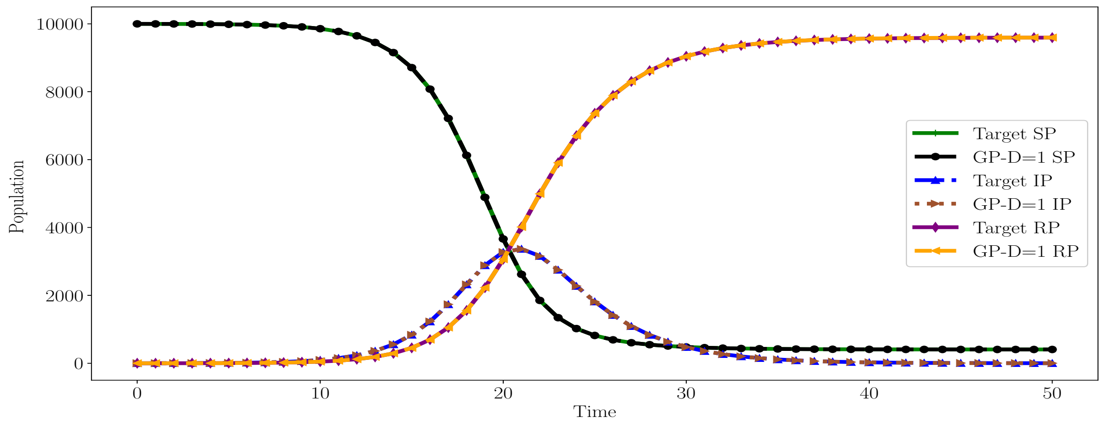


Figure 10. Illustration of Case 2 target outputs, susceptible population (SP), infected population (IP) and recovery population (RP), and the two generated outputs from the corresponding inferred models for Setup 2 with genetic programming depth controller.

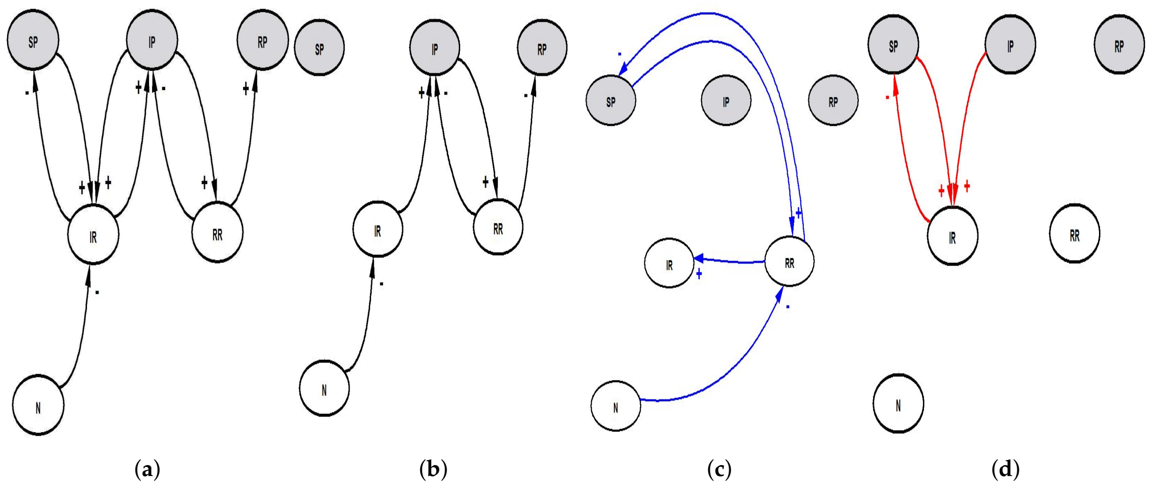
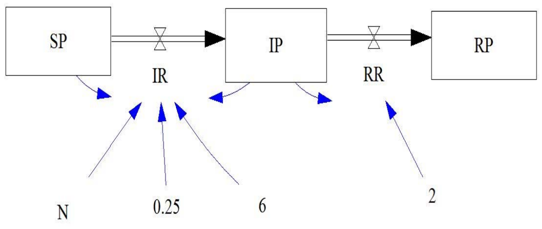
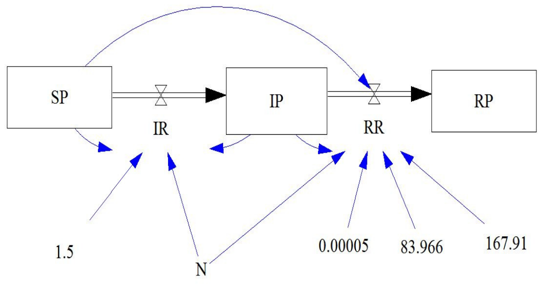


Figure 11. Case 2 inferred CLD by showing the correctly predicted links (**b**); additional links (**c**); and missing links (**d**) compared to the target CLD (**a**)—Setup 2.



(**a**) Target SFD.



(**b**) Inferred SFD.

Figure 12. Case 2 inferred SFD vs. target one—Setup 2.

For Setup 2, execution times for cases 1 and 2 are around 4 and 5 h, respectively. Since CLD and equation structures are inferred with parameter estimation, both number of stocks and relationship complexity influence execution time. In addition, the execution time for this Setup is longer than Setup 1 which makes sense. From the perspective of modeling time, we think this an acceptable time to infer such models. It is probably much quicker to generate CLD, SFD, equations and their parameters, with the ability to navigate and produce a large number of models, than to develop these models manually. However, the human-based modeling has the benefit of producing more meaningful and structurally valid models than the CI-based methods can do. In order to determine whether the execution time in this Setup is reliable, it should be compared with the time taken by a human modeler to come up with these case models.

Table 5. Runtime for proposed computational intelligence methods in generating system dynamics models for each case with different setups.

Case Studies	Setup 1	Setup 2
Case 1	6.2 min	4 h
Case 2	6.9 min	5 h

4.4. Limitations

The concept of structural validity of method under development generated models is not applied in this paper, where graphical or topological error measures are only applied to compare the generated models against the target synthetic models. The structural credibility of models was assessed by asking the following questions: Do the equations make causal sense? Do they have a real-life causal meaning? Do they possess unit consistency? Do units of the equations naturally yield the units of the output variable, without adding any dummy 'fudging' coefficients? Do the equations yield logical extreme values, under extreme input values? Are the equations robust/valid under high and low parameter values?

The GP ensemble used to generate the model equations suffers from two main issues: first, the predefined functional sets that list the mathematical operators and functions used to build the GP trees. In our experimental setups, we defined these sets according to our knowledge of case study models nature. However, this assumption is a weak point in the proposed GP ensemble which could be fixed either by defining functional sets that contain a large collection of algebraic and nonlinear functions, or by learning these sets during the evolution process. Second, the defined fitness function relies on the output error between the target behavior and generated behavior. The main purpose of the proposed GP ensemble is to generate models that have valid structures and not just models that generate behavior similar to that of the target. As we see from the results, for Case 2 in particular, two different model structures inferred in both Setup 1 and 2 can generate the same behavior despite having different equation structures. The main purpose of using ensemble learning method is to avoid inferring the whole system equations all at once given the time-series observations. This could lead to generate too large equations for the sake of matching the target ones since the fitness function is mainly derived by output error. This problem is a well-known issue in the literature called bloating [29]. Furthermore, the balance between the model structure and generated behaviors should be preserved as much as we can to provide SD modelers with model equations that make sense at least in terms of equations size.

Our proposed methodology relies mainly on the CLDs to generate both SFD and equations. In Setup 1, we assume that target CLD is given and based on this we generate the equations. In Setup 2, we learn CLD and underlying equations simultaneously, and, after the SFD is built, by annotating each variable in the CLD. Although CLDs play an important role in the SD modeling process, such as enabling the smooth transition to the final SFDs and early engagement with system stakeholders, they are not in themselves the complete SD models. From the perspective of the SD modeling process, it makes more sense to use SFDs instead of CLDs to generate the equations, where CLD can be generated easily from SFD for presentation purposes. The reason for choosing to learn the model equations based on the CLDs and not SFDs is due to the smaller search space needed to infer CLDs than to infer SFDs. With CLDs, we are only interested in the links between variables, whereas, with SFDs, we aim to identify the variable types (e.g., stock, in/out flow, auxiliary) and the links between them. This could make the overall inferring process more complicated and costly.

The integration of the SA and GP ensemble algorithm to generate both CLD and equations simultaneously is still a costly process. The reason is that both processes are nested and the quality of the CLD depends on the generated equations that should be simulated first to generate behavior. This nested process could be avoided by determining how to evaluate the CLD independently, before giving it to the GP ensemble to generate the underlying equations. Furthermore, we need to use a wide range of case studies that scale from simpler to more complicated ones to see how the proposed CI methods' calculation time scale for large scale cases.

We assume that the stocks are known and that time series observations will be available for all of them. This assumption is necessary for two purposes: for the GP ensemble methodology which is used to generate the mathematical representation, with differential equations, of system stock variables; and for SFD inferring where the identification of the remaining system variables starts from knowing the stocks. However, the system observations could refer to stock variables or to any other variables, such as flows or auxiliaries. In addition, the availability of the observations for all system variables, regardless of their types, is not always known.

5. Conclusions

Our ability to store and process increasing amounts of data and then to utilize CI methods to turn this data into useful information is driving a revolution across the SD field. SD is a powerful tool used in the modeling of complex dynamical systems. It is extensively applied to dynamic phenomena emerging in industrial, economic, social, ecological and transportation systems, as well as being used for a variety of tasks, such as policy analysis and design, learning and training, decision-making and forecasting. This paper contributes to a growing field of interest that aims to facilitate and improve the efficiency of the SD modeling process by developing a CI-based modeling support system to generate SD models from system observations. The core component of the support system is the learning engine which builds upon different CI methods. These can be invoked in two different inferring modes: inferring only the underlying equations and their parameters using the GP ensemble; and inferring CLDs, SFDs, equations and estimating their parameters by the integration of the GP ensemble and SA algorithms. Based on the inferred CLD and system equations, the SFD was created by applying a fixed rule-based system to identify the variable types as either inflows, outflows, auxiliaries or parameters.

To illustrate how the support system could be applied to generate SD models close to target models, we chose two synthetic reality cases from the epidemics domain. Two main setups were applied, Setup 1 for generating the system equations and estimating their parameters given the target CLD, and Setup 2 for generating CLD, SFD, equations and estimating the model parameters. The experimental results from Setup 1 show the ability of the GP ensemble to match the target equations and generate the same behaviors for Case 2. Although the generated equations for Case 1 have missing terms, these inferred equations are able to generate a very close match to the target behavior. These differences in the generated equations are affected by the depth controlled parameter that controls the GP tree depth. It has the advantage of preventing the generated equations from growing and at the same time generates the desired behavior since the fitness function is mainly derived from the output error between the generated and target behaviors. The results from Setup 2 show that the inferred CLDs, SFDs and equations have some similarities to the target ones. There are additional and missing links in the generated CLD and SFD, and additional terms in the inferred equations. The inferred structures generate behaviors which match those of the target because the GP fitness function is derived from the output error. In order to guide the GP inferring process to generate valid structures in addition to minimizing the output error, a structural measure needs to be integrated with the output error fitness. This structural measure should be inspired and derived from structural validation tests applied to SD models built by SD practitioners and modelers. The performance of the GP ensemble is verified with specific conditions, where all system stocks, their observations and the mathematical nature of the target equations are known. Changing one of these conditions will have an effect on the performance of GP ensemble, in addition to the uncertainty in the system observations which could also affect the robustness of GP algorithm. Regarding the conducted runtime analysis, the scale of the problem affects the evolution process of the CI algorithms—specifically if we want to infer different model parts synchronously such as CLDs, SFDs or equations which are tightly coupled through the SD modeling process.

The proposed support system in this paper shows great promise to support SD modeling process which will enable the modelers to save time and effort in building an SD model by providing the SD modeler with an inferred model where the modeler could check and validate. Several directions can be taken to extend this work:

1. Most pressing is the need to work towards enhancing the ability of the methods to generate models with minimal structures that can characterize the data. The methods under development should be robust and scalable with the ability to handle big data.
2. It is crucial to integrate the necessary semantic domain knowledge, about the system or problem of interest, with the CI-based methods to generate valid structures. This will add a new dimension to the capability of the CI-methods for generating valuable models, especially for real-life systems where we do not have any idea about how the target model looks.
3. Identification of which variables to include in the model, as part of the support system's inference engine, is challenging because, for each variable set selected, a SD model should be built and simulated to evaluate those selected. Therefore, the efficiency of the modeling process would be greatly enhanced by finding a method which bypassed the need to build the whole SD model.
4. Generating an SD model without prior knowledge of the types of variables and only limited observations is a challenging task too. However, the ability to handle these conditions since they are common in real world applications is an important feature to be added to the support system. Generating SD models under these conditions will require not only a search for the set of causal and mathematical relationships, but a search for the types of variables and for the mathematical equations for those variables that do not have any observations.

Supplementary Materials: List of acronyms, Case studies simulation model description and python code, Computational intelligence algorithms pseudo code, and User manual for system dynamics modeling support system, are available online at http://www.mdpi.com/2079-8954/7/4/47/s1.

Author Contributions: Conceptualization, H.A.; methodology, H.A.; software, H.A.; validation, H.A. and K.S; writing–original draft preparation, H.A.; writing–review and editing, H.A. and K.S.; supervision, K.S.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mohammadifardi, H.; Knight, M.A.; Unger, A.A. Sustainability Assessment of Asset Management Decisions for Wastewater Infrastructure Systems—Implementation of a System Dynamics Model. *Systems* **2019**, *7*, 34. [CrossRef]
2. Suprun, E.; Sahin, O.; Stewart, R.; Panuwatwanich, K.; Shcherbachenko, Y. An Integrated Participatory Systems Modelling Approach: Application to Construction Innovation. *Systems* **2018**, *6*, 33. [CrossRef]
3. Reinker, M.; Gralla, E. A System Dynamics Model of the Adoption of Improved Agricultural Inputs in Uganda, with Insights for Systems Approaches to Development. *Systems* **2018**, *6*, 31. [CrossRef]
4. Winch, G.W.; Arthur, D.J. User-parameterised generic models: A solution to the conundrum of modelling access for SMEs? *Syst. Dyn. Rev.* **2002**, *18*, 339–357. [CrossRef]
5. Kanninga, P. Simulation Model Development, The Devil Is in the Detail! Ph.D. Thesis, Delft University of Technology, TU Delft, The Netherlands, 2008.
6. Pruyt, E.; Cunningham, S.; Kwakkel, J.; De Bruijn, J. From data-poor to data-rich: System dynamics in the era of big data. In Proceedings of the 32nd International Conference of the System Dynamics Society, Delft, The Netherlands, 20–24 July 2014; System Dynamics Society: Albany, NY, USA, 2014.
7. Bourguet, R.E.; Soto, R. Qualitative knowledge acquisition using fuzzy logic and system dynamics. In Proceedings of the 20th International Conference of the System Dynamics Society, Palermo, Italy, 28 July–1 August 2002; System Dynamics Society: Albany, NY, USA, 2002.
8. Ho, Y.F.; Wang, H.L. Applying fuzzy Delphi method to select the variables of a sustainable urban system dynamics model. In Proceedings of the 26th International Conference of the System Dynamics Society, Athens, Greece, 20–24 July 2008; System Dynamics Society: Albany, NY, USA, 2008.
9. Abdelbari, H.; Shafi, K. A Computational Intelligence-based Method to Learn Causal Loop Diagram-like Structures from Observed Data. *Syst. Dyn. Rev.* **2017**, *33*, 3-33. [CrossRef]
10. Abdelbari, H.; Shafi, K. Optimizing a Constrained Echo State Network using Evolutionary Algorithms for Learning Mental Models of complex dynamical systems. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 4735–4742.
11. Abdelbari, H.; Shafi, K. Learning Structures of Conceptual Models from Observed Dynamics using Evolutionary Echo State Networks. *J. Artif. Intell. Soft Comput. Res.* **2018**, *8*, 133–154. [CrossRef]
12. Chen, Y.T.; Tu, Y.M.; Jeng, B. A machine learning approach to policy optimization in system dynamics models. *Syst. Res. Behav. Sci.* **2011**, *28*, 369–390. [CrossRef]
13. Abdelbari, H.; Elsawah, S.; Shafi, K. Model Learning using Genetic Programming under Full and Parial System Information Conditions. In Proceedings of the 33rd International Conference of the System Dynamics Society, Cambridge, MA, USA, 19–23 July 2015; System Dynamics Society: Albany, NY, USA, 2015.
14. Abdelbari, H.; Shafi, K. A Genetic Programming Ensemble Method for Learning Dynamical System Models. In Proceedings of the 8th International Conference on Computer Modeling and Simulation, Canberra, Australia, 20–23 January 2017.
15. North, M.; Sydelko, P.; Martinez-Moyano, I. Structurally Evolving System Dynamics Models Using Genetic Algorithms. In Proceedings of the 33rd International Conference of the System Dynamics Society, Cambridge, MA, USA, 19–23 July 2015, System Dynamics Society: Albany, NY, USA, 2015.
16. Drobek, M.; Gilani, W.; Molka, T.; Soban, D. Automated equation formulation for causal loop diagrams. *Lect. Notes Bus. Inf. Process.* **2015**, *208*, 38–49.
17. Yücel, G.; Barlas, Y. Automated parameter specification in dynamic feedback models based on behavior pattern features. *Syst. Dyn. Rev.* **2011**, *27*, 195–215. [CrossRef]
18. Wu, Z.; Xu, J. Predicting and optimization of energy consumption using system dynamics-fuzzy multiple objective programming in world heritage areas. *Energy* **2013**, *49*, 19–31. [CrossRef]
19. Struben, J.; Sterman, J.; Keith, D. Parameter Estimation through Maximum Likelihood and Bootstrapping Methods. In *Analytical Methods for Dynamic Modelers*; MIT Press: Cambridge, MA, USA, 2015; pp. 3–38.
20. Liu, H.; Howley, E.; Duggan, J. Optimisation of the Beer Distribution Game with complex customer demand patterns, In Proceedings of the 2009 Congress on Evolutionary Computation. Trondheim, Norway, 18–21 May 2009.

21. Phelan, M.; McGarraghy, S. Mitigating the bullwhip effect in supply chains using grammatical evolution. In Proceedings of the 25th International Conference of the System Dynamics Society, Boston, MA, USA, 29 July–2 August 2007; System Dynamics Society: Albany, NY, USA, 2007.
22. Graham, A.K.; Ariza, C.A. Dynamic, hard and strategic questions: Using optimization to answer a marketing resource allocation question. *Syst. Dyn. Rev.* **2003**, *19*, 27–46. [CrossRef]
23. Rahmandad, H.; Oliva, R.; Osgood, N.D.; Richardson, G. Using Decision Trees to Value Managerial Real Options. In *Analytical Methods for Dynamic Modelers*; MIT Press: Cambridge, MA, USA, 2015; pp. 307–336.
24. Rahmandad, H.; Spiteri, R.J. Modeling Comparing Actors using Differential Games. In *Analytical Methods for Dynamic Modelers*; MIT Press: Cambridge, MA, USA, 2015; pp. 373–404.
25. Miller, J.H. Active nonlinear tests (ANTs) of complex simulation models. *Manag. Sci.* **1998**, *44*, 820–830. [CrossRef]
26. Yücel, G.; Barlas, Y. Pattern recognition for model testing, calibration, and behavior analysis. In *Analytical Methods for Dynamic Modelers*; MIT Press: Cambridge, MA, USA, 2015; pp. 173–206.
27. Pei, W. Fuzzy Evaluation on the Validity of System Dynamics Models. In *Computer-Based Management of Complex Systems*; Springer: Berlin/Heidelberg, Germany, 1989; pp. 271–276.
28. Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; MIT Press: Cambridge, MA, USA, 1992; Volume 1.
29. O'Neill, M.; Vanneschi, L.; Gustafson, S.; Banzhaf, W. Open issues in genetic programming. *Genet. Program. Evolvable Mach.* **2010**, *11*, 339–363. [CrossRef]
30. Quade, M.; Abel, M.; Shafi, K.; Niven, R.K.; Noack, B.R. Prediction of Dynamical Systems by Symbolic Regression. *Phys. Rev. E* **2016**, *94*, 012214. [CrossRef] [PubMed]
31. Soule, T. Code Growth in Genetic Programming. Ph.D. Thesis, University of Idaho, Moscow, ID, USA, 1998.
32. Miller, J.F. Cartesian Genetic Programming. In *Cartesian Genetic Programming*; Springer: Berlin, Germany, 2011; pp. 17–34.
33. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
34. Batista, G.E.; Wang, X.; Keogh, E.J. A complexity-invariant distance measure for time series. In Proceedings of the 2011 International Conference on Data Mining, Mesa, AZ, USA, 28–30 April 2011; pp. 699–710.
35. Keogh, E.; Wei, L.; Xi, X.; Vlachos, M.; Lee, S.H.; Protopapas, P. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures. *VLDB J.* **2009**, *18*, 611–630. [CrossRef]
36. Keogh, E. Efficiently finding arbitrarily scaled patterns in massive time series databases. In Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery, Cavtat-Dubrovnik, Croatia, 22–26 September 2003; Springer: Berlin, Germany, 2003, pp. 253–265.
37. Faloutsos, C.; Ranganathan, M.; Manolopoulos, Y. *Fast Subsequence Matching in Time-Series Databases*; ACM: New York, NY, USA, 1994; Volume 23.
38. Oliva, R. Model structure analysis through graph theory: Partition heuristics and feedback structure decomposition. *Syst. Dyn. Rev.* **2004**, *20*, 313. [CrossRef]
39. Takahashi, Y. Stock Flow Diagram Making with Incomplete Information about Time Properties of Variables. In Proceedings of the 24th International Conference of the System Dynamics Society, Nijmegen, The Netherlands, 23–27 July 2006; System Dynamics Society: Albany, NY, USA, 2006.
40. Forrester, J.W. System Dynamics Self Study—MIT OpenCourseWare. Available online: http://ocw.mit.edu/courses/sloan-school-of-management/15-988-system-dynamics-self-study-fall-1998-spring-1999/ (accessed on 19 July 2017).
41. Sterman, J.D. *Business Dynamics: Systems Thinking and Modeling for a Complex World*; Irwin/McGraw-Hill: Boston, MA, USA, 2000; Volume 19.