Online Supplementary Material for:

A System Dynamics Modeling Support System based on Computational Intelligence

Contents

1 Background **2**
1.1 List of Acronyms . . . 2

2 Case Studies **2**
2.1 Case 1 . . . 2
2.1.1 Model Variables Description . . . 2
2.1.2 Model Equations . . . 3
2.1.3 Simulation Model Code and Generated Behavior . . . 3
2.2 Case 2 . . . 4
2.2.1 Model Variables Description . . . 4
2.2.2 Model Equations . . . 5
2.2.3 Simulation Model Code and Generated Behavior . . . 5

3 CI Algorithms **7**

4 User Manual for System Dynamics Modeling Support System **9**
4.1 Support System Features . . . 9
4.2 Prerequisites and How to Run the Support System . . . 10
4.3 Support System GUI . . . 10
4.3.1 Main Screen . . . 10
4.3.2 Inputs Screen . . . 13
4.3.3 CLD Learning Screen . . . 22
4.3.4 Equations Learning and Parameters Estimation Given Manually Created CLD Screen . . . 24
4.3.5 CLD, SFD and Equations Learning Screen . . . 25

1 Background

1.1 List of Acronyms

Table 1: List of acronyms related to the computational intelligence methods.

Acronym	Description
FL	Logic operation method based on many valued logic rather than binary logic.
FMOP	Decision making approach based on the desirable features of compromise programming and fuzzy set theory.
FFNN	Type of artificial neural networks where each neuron or network node in one layer has directed forward connections to the neurons in the next layer.
RNN	Type of artificial neural networks which allows loops and cycles among network neurons.
ESN	Recent type of RNNs that work with fixed network weights and a relatively straightforward training procedure.
GA	Stochastic optimization method inspired from a genetic natural evolution procedure of living beings.
GP	Population-based evolutionary algorithm, representing individual solutions as programs instead of a string of bits, as in a GA.
CGP	Special, flexible and highly efficient type of GP algorithm that encodes a graph representation of a computer program and uses only point mutation operator without an applied crossover operator.
GE	Evolutionary search algorithm, similar to GP typically used to generate programs with syntax defined through a grammar.
PSO	Search method inspired from the behavior of flocking birds.
SA	Global optimization algorithm inspired from the cooling process of a material.

2 Case Studies

2.1 Case 1

2.1.1 Model Variables Description

Table 2 shows the model list of variables by providing the variable name, the abbreviation that will be used to refer to this variable and the variable type.

Table 2: Case 1 variables names, abbreviations and types.

Name	**Abbreviation**	**Type**
Healthy People	HP	Stock
Sick People	SP	Stock
Recovery Rate	RR	Flow
Catching Illness	CI	Flow
Probability of Contact with Sick People	PCSP	Auxiliary
Duration of Illness	DI	Parameter
Probability of Catching Illness	PCI	Parameter
Population Interaction	PI	Parameter

2.1.2 Model Equations

$$\begin{aligned} \frac{dHP}{dt} &= RR - CI \\ \frac{dSP}{dt} &= CI - RR \\ CI &= HP * PCI * PCSP * PI \\ RR &= \frac{SP}{DI} \\ HP_0 &= 99 \\ SP_0 &= 1 \\ DI &= 0.5 \\ PCI &= 0.5 \\ PI &= 10 \end{aligned} \tag{1}$$

By replacing the flows, auxiliaries and parameters values directly into stock variables equations, the target model equations that describe the stocks behavior will be:

$$\begin{aligned} \frac{dHP}{dt} &= (2 * SP) - (\frac{5 * HP * SP}{HP + SP}) \\ \frac{dSP}{dt} &= (\frac{5 * HP * SP}{HP + SP}) - (2 * SP) \end{aligned} \tag{2}$$

2.1.3 Simulation Model Code and Generated Behavior

In this section, the simulation model source code is provided which used to generate a sample of 50 simulation runs with Case 1. These generated samples are used in the experiments. The code is written in python as follows:

```
import numpy as np
import matplotlib.pyplot as plt

def Epidemics(HP, SP):
    HP_dot = (2*SP) - ((5*HP*SP)/(HP+SP))
```

```
    SP_dot = ((5*HP*SP)/(HP+SP)) - (2*SP)
    return HP_dot, SP_dot

dt = 0.0625
stepCnt = 80

HP = np.empty((stepCnt+1,))
SP = np.empty((stepCnt+1,))

HP[0], SP[0] = (99.0, 1.0)

for i in range(stepCnt):
    HP_dot, SP_dot = Epidemics(HP[i], SP[i])
    HP[i+1] = HP[i] + (HP_dot * dt)
    SP[i+1] = SP[i] + (SP_dot * dt)

np.savetxt('c:\HP.txt', HP, delimiter='\n')
np.savetxt('c:\SP.txt', SP, delimiter='\n')

plt.plot(HP, color="red", label="HP")
plt.plot(SP, color="blue", label="SP")
plt.legend()
plt.show()
```

2.2 Case 2

2.2.1 Model Variables Description

Table 3 shows the model list of variables by providing the variable name, the abbreviation that will be used to refer to this variable and the variable type.

Table 3: Case 2 variables names, abbreviations and types.

Name	**Abbreviation**	**Type**
Susceptible Population	SP	Stock
Infected Population	IP	Stock
Recovered Population	RP	Stock
Infection Rate	IR	Flow
Recovery Rate	RR	Flow
Total Number of Population	N	Input
infectivity	i	Parameter
contacts	c	Parameter
duration	d	Parameter

2.2.2 Model Equations

$$
\begin{aligned}
\frac{dSP}{dt} &= -IR \\
\frac{dIP}{dt} &= IR - RR \\
\frac{dRP}{dt} &= RR \\
IR &= (c * i * SP)(\frac{IP}{N}) = \frac{c * i * SP * IP}{N} \\
RR &= \frac{IP}{d} \\
SP_0 &= 9999 \\
IP_0 &= 1 \\
RP_0 &= 0 \\
i &= 0.25 \\
c &= 6 \\
d &= 2 \\
N &= 10000
\end{aligned}
\tag{3}
$$

By replacing the flows, auxiliaries and parameters values directly into stock variables equations, the target model equations that describe the stocks behavior will be:

$$
\begin{aligned}
\frac{dSP}{dt} &= -\frac{1.5 * SP * IP}{N} \\
\frac{dIP}{dt} &= \frac{1.5 * SP * IP}{N} - 0.5 * IP \\
\frac{dRP}{dt} &= 0.5 * IP
\end{aligned}
\tag{4}
$$

2.2.3 Simulation Model Code and Generated Behavior

In this section, the simulation model source code is provided which used to generate a sample of 50 simulation runs with Case 1. These generated samples are used in the experiments. The code is written in python as follows:

```
import numpy as np
import matplotlib.pyplot as plt

def SIR(SP, IP, RP, N=10000):
    SP_dot = (-1.5*SP*IP)/N
    IP_dot = ((1.5*SP*IP)/N) - (0.5*IP)
    RP_dot = (0.5*IP)
    return SP_dot, IP_dot, RP_dot

dt = 0.58
stepCnt = 50

SPs = np.empty((stepCnt+1,))
```

```
IPs = np.empty((stepCnt+1,))
RPs = np.empty((stepCnt+1,))

SPs[0], IPs[0], RPs[0] = (9999.0, 1.0, 0.0)

for i in range(stepCnt):
    SP_dot, IP_dot, RP_dot = SIR(SPs[i], IPs[i], RPs[i])
    SPs[i+1] = SPs[i] + (SP_dot * dt)
    IPs[i+1] = IPs[i] + (IP_dot * dt)
    RPs[i+1] = RPs[i] + (RP_dot * dt)

np.savetxt('c:\SP.txt', SPs, delimiter='\n')
np.savetxt('c:\IP.txt', IPs, delimiter='\n')
np.savetxt('c:\RP.txt', RPs, delimiter='\n')

plt.plot(SPs, color="red", label="SP")
plt.plot(IPs, color="blue", label="IP")
plt.plot(RPs, color="green", label="RP")
plt.legend()
plt.show()
```

3 CI Algorithms

Algorithm 1 Ensemble member Cartesian GP for equation learning

Input: population size (P_{size}), terminal set (T_{set}), functional set (F_{set}), mutation probability (P_m)
Output: S_{best}
$P \leftarrow CreateInitialPopulation(P_{size}, T_{set}, F_{set})$
$EvaluatePopulation(P)$
$S_{best} \leftarrow GetBestSolution(P)$
while $\neg StoppingCriteria()$ **do**
  $P_{temp} \leftarrow \emptyset$
  while Size(P_{temp}) $< P_{size}$ **do**
    $S_{parent} \leftarrow Selection(P)$
    $S_{child} \leftarrow Mutation(S_{parent})$
    if $\epsilon(S_{child})leq\epsilon(S_{parent})$ **then**
      $P_{temp} \leftarrow S_{child}$
    else
      $P_{temp} \leftarrow S_{parent}$
    end if
  end while
  $EvaluatePopulation(P_{temp})$
  $S_{best} \leftarrow GetBestSolution(P_{temp})$
  $P \leftarrow P_{temp}$
  $S_{best} \leftarrow GetBestSolution(P)$
end while
return S_{best}

Algorithm 2 Ensemble member SA for parameters estimation

Input: maximum iterations I, maximum sub iterations I_{sub}, temperature T, cooling constant α, perturbation constant δ
Output: S_{best}
$S_{new} \leftarrow GetParametersValuesFromGPTree()$
$S_{best} \leftarrow S_{new}$
for $i \leftarrow 1$ to I **do**
  for $j \leftarrow 1$ to I_{sub} **do**
    $S_i \leftarrow CreateNewParametersValues(S_{new}, \delta)$
    if $\epsilon(S_i) < \epsilon(S_{new})$ **then**
      $S_{new} \leftarrow S_i$
    else
      $\Delta E \leftarrow \epsilon(S_{new}) - \epsilon(S_i)$
      $P \leftarrow e^{\frac{-\Delta E}{T}}$
      if $rand(0,1) \leq P$ **then**
        $S_{new} \leftarrow S_i$
      end if
    end if
    if $\epsilon(S_{new}) < \epsilon(S_{best})$ **then**
      $S_{best} \leftarrow S_{new}$
    end if
  end for
  $T \leftarrow \alpha T$
end for
return S_{best}

Algorithm 3 SA+GP ensemble for CLD and equations learning with parameters estimation

Input: maximum iterations I, maximum sub iterations I_{sub}, temperature T, cooling constant α, links flipping probability β, structure sparsity ρ
Output: S_{best}
$S_{new} \leftarrow CreateInitialCLD(\rho)$
$S_{best} \leftarrow S_{new}$
for $i \leftarrow 1$ to I **do**
  for $j \leftarrow 1$ to I_{sub} **do**
    $S_i \leftarrow CreateNewCLD(S_{new}, \beta)$
    if $\epsilon(S_i) < \epsilon(S_{new})$ **then**
      $S_{new} \leftarrow S_i$
    else
      $\Delta E \leftarrow \epsilon(S_{new}) - \epsilon(S_i)$
      $P \leftarrow e^{\frac{-\Delta E}{T}}$
      if $rand(0,1) \leq P$ **then**
        $S_{new} \leftarrow S_i$
      end if
    end if
    if $\epsilon(S_{new}) < \epsilon(S_{best})$ **then**
      $S_{best} \leftarrow S_{new}$
    end if
  end for
  $T \leftarrow \alpha T$
end for
return S_{best}

4 User Manual for System Dynamics Modeling Support System

In this section, a user manual guide for the developed support system is provided. The target end users of this tool are SD practitioners, expert modelers and scholars where the minimal knowledge about SD is required. The support system provides a friendly GUI interface which receives all the necessary information about the system of interest and learn system dynamics models including CLDs, system stocks differential equations and model calibration and types of variables in SFD terminologies.

4.1 Support System Features

The main aim of the support system is to learn SD model artifacts automatically from system observations toward saving the time and effort spent during the modeling process and enhancing the overall performance and accuracy of the developed models. The necessary inputs required from the end user are as follows:

- List of system key variables.
- Time series observations for system output variables and optionally for input variables.

- The CI algorithms controlling parameters.

Given these inputs, the support system applies different CI algorithms to learn SD models in three learning modes:

- Learning CLD structures.
- Learning system differential equations and estimate model parameters given a manually created CLD structure.
- Learning the CLD and system differential equations and model calibration and identifying variables types in SFD terminologies.

4.2 Prerequisites and How to Run the Support System

The support system tool directory contains the following files and sub directories as follows:

- jars (sub directory)
- README.TXT (text file)
- Tool_Run.bat (batch file)

where *README.TXT* file contains the necessary instructions and prerequisites to run the support system, *jars* sub directory contains the java archived files (libraries) used by the tool and *Tool_Run.bat* is the batch file used to run the system. Before running the the tool, the Java SE development kit version 1.8 should be installed on the machine. The link to download this kit is provided in the *README.TXT* file. After installing the kit, the tool can be run by double clicking the batch file.

4.3 Support System GUI

The support system GUI consists of five main screens: the main, system inputs, CLD learning, equations learning and parameters estimation, and SD model learning as shown in Figures 1a, 1b, 1c, 1d and 1e, respectively.

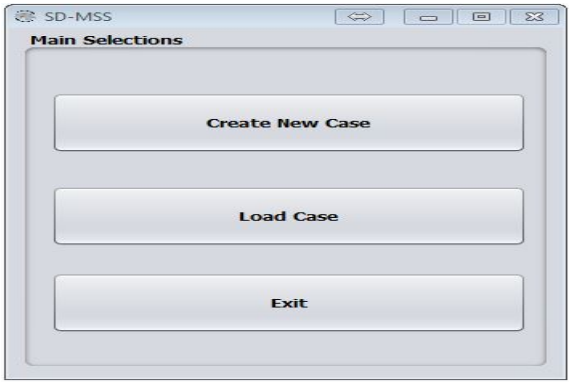
4.3.1 Main Screen

The main screen consists of three main buttons: *Create New Case*, *Load Case* and *Exit* as shown in Figure 1a. *Create New Case* button is used to create a new case study directory, *Load Case* button is used to load an existing case study, and *Exit* button is used to exit from the tool.

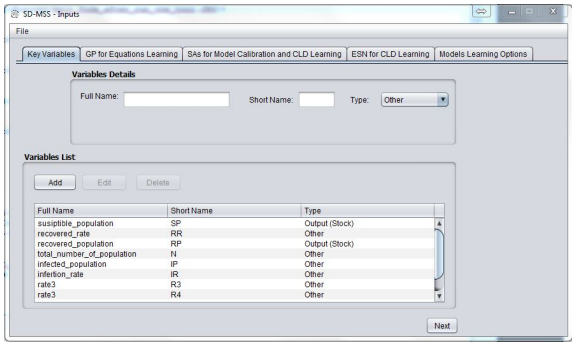
4.3.1.1 Create New Case By clicking on the *Create New Case* button, an open dialog box is shown to let the user select the new case directory as shown in Figure 2. The selected directory for a new case should be empty and created by the user.

If the user selected a non-empty directory, a message box will be shown (Figure 3) to notify the user about making sure to select an empty directory for the new case. Once the case directory is selected, the inputs screen is viewed as shown in Figure 1b.

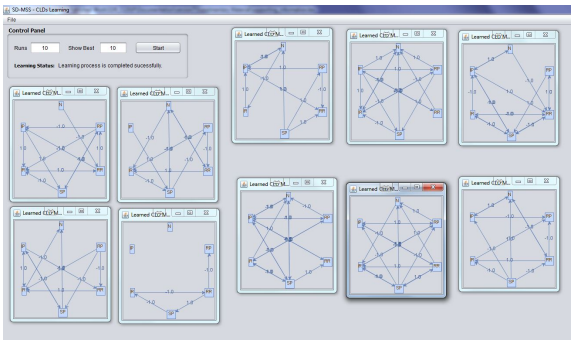
4.3.1.2 Load Case To load a previously created case, the user can click on the *Load Case* button which shows an open dialog box to select the case directory as shown in Figure 2. The selected case directory should be created before using the tool. In case of selecting an empty directory, a notification message will be shown to the user (Figure 4).



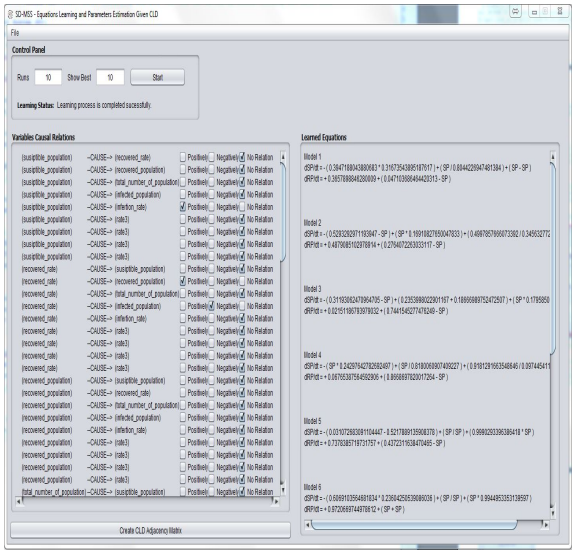
(a) Main screen



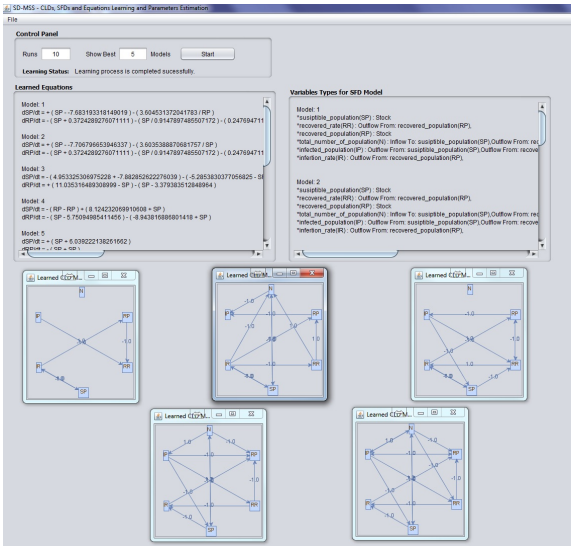
(b) Inputs screen



(c) CLD learning screen



(d) Equations learning and parameters estimation screen



(e) SD model end-to-end learning screen.

Figure 1: Snapshots of the support system tool GUI screens.

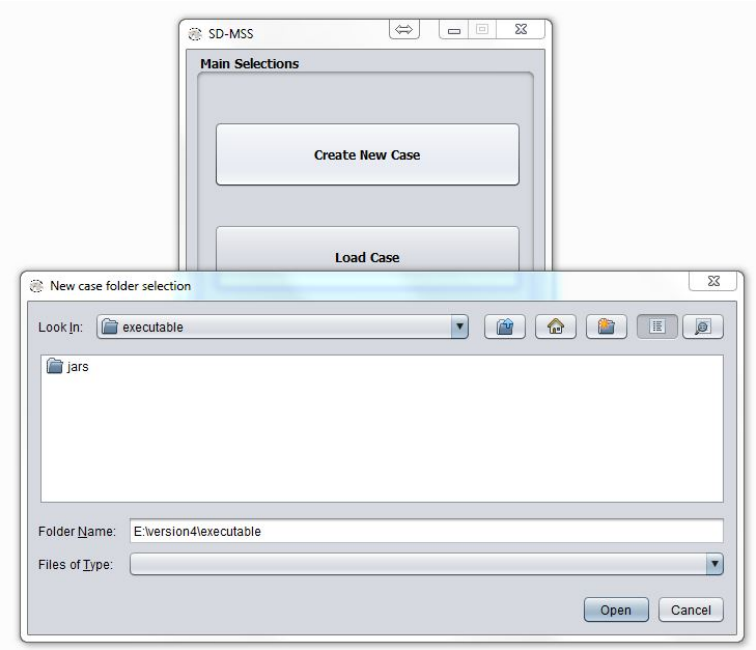


Figure 2: A snapshot of the new case dialog box to select the case directory.

Figure 3: A snapshot of the message viewed to the user in case of selecting a non-empty directory for new created case.

Figure 4: A snapshot of the message viewed to the user in case of selecting an empty directory when loading a case.

4.3.2 Inputs Screen

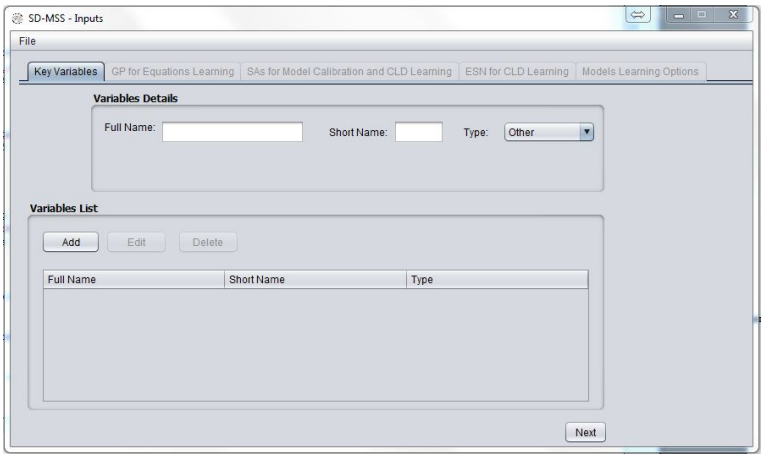
This screen is viewed directly after either creating a new case or loading an existed one. It lets the end-user provides the required information about the system of interest. Figures 5a and 5b show this screen when a new case is created or a previously created one is loaded, respectively.

This form has five main tabs: the variables tab (Figure 6a) for adding the required information about the system key variables and upload their time series observations, the GP algorithm tab (Figure 6b) for identifying the controlling parameters for the GP algorithm, the SA algorithms tab (Figure 6c) for identifying the controlling parameters for SA algorithms, the ESN algorithm tab (Figure 6d) for identifying the controlling parameters for ESN algorithm and learning options tab (Figure 6e) for selecting the learning mode.

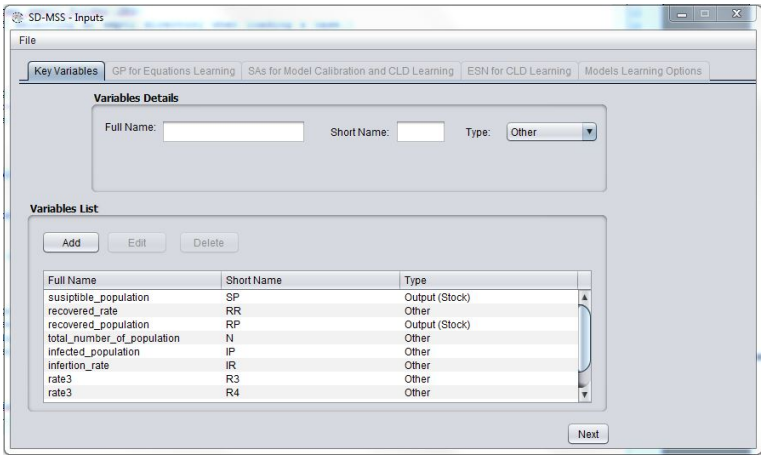
To return into the main screen, the back menu item under the file main menu is clicked as shown in Figure 7.

4.3.2.1 Variables Tab In this tab, the user can add, edit or delete system variables.

Adding New Variable To add a new variable, the variable fields should be filled. Each variable has three main fields: full name, short name and type. Full name describes the variable, while the short name is an abbreviation used to refer to this variables when learning the equations and the CLD structure. Variable type could be either output, input or other. The name fields should be strings, without spaces and cannot by empty. In case of selecting an output or input types for a variable, an upload button becomes

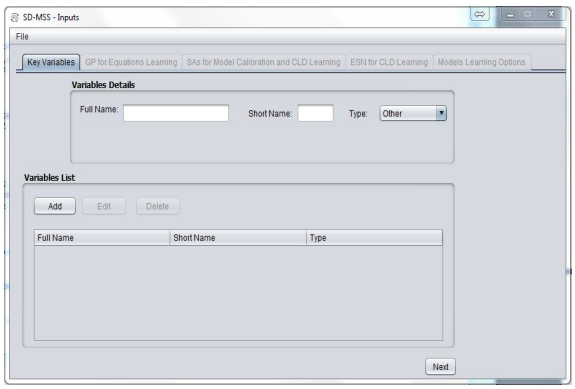


(a) Creating a new case

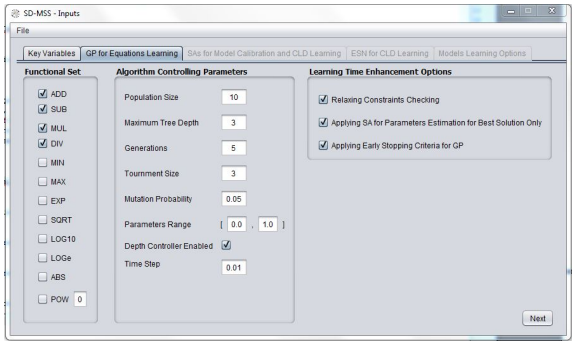


(b) Loading a case

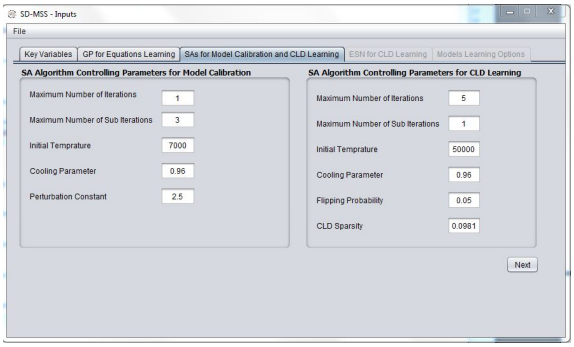
Figure 5: A snapshots of the inputs screen when a new case is created or loaded.



(a) Variables tab

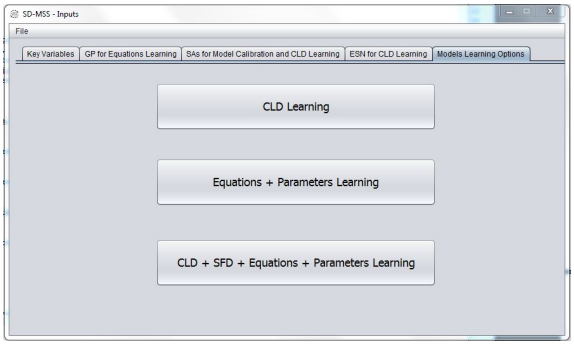


(b) GP tab



(c) SA tab

(d) ESN tab



(e) Learning options tab

Figure 6: Snapshots of inputs screen tabs.

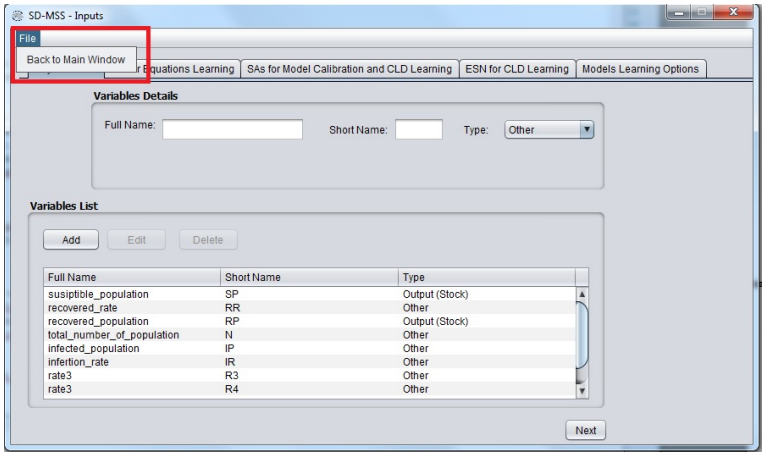


Figure 7: Inputs window back menu.

visible to upload the time series data corresponding to this variable as shown in Figure 8. This upload button disappeared in case of selecting other for variable type.

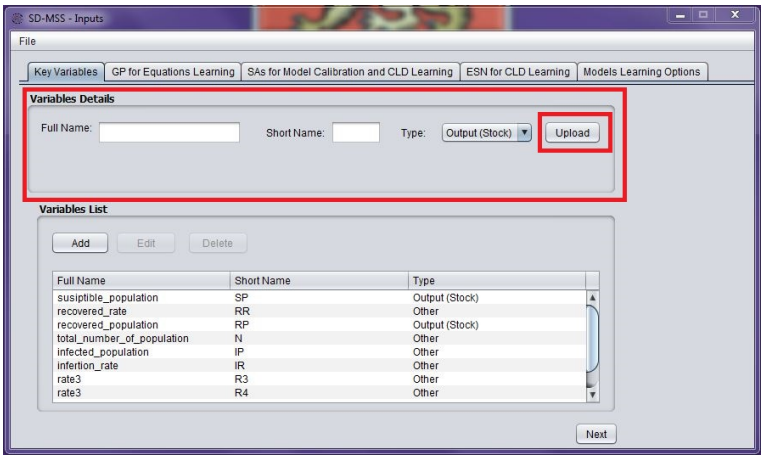


Figure 8: A snapshot of the variables tab when the upload button is activated to be used in uploading the time series data for input and output variable types.

The uploaded time series files for output and input variables should be in text files with numerical representation and listed vertically in the file. Figure 9 shows an example for a times series text file. If the user selects an invalid time series text file that has non-numerical values or they are not arranged line by line, a pop up message will be viewed as shown in Figure 10.

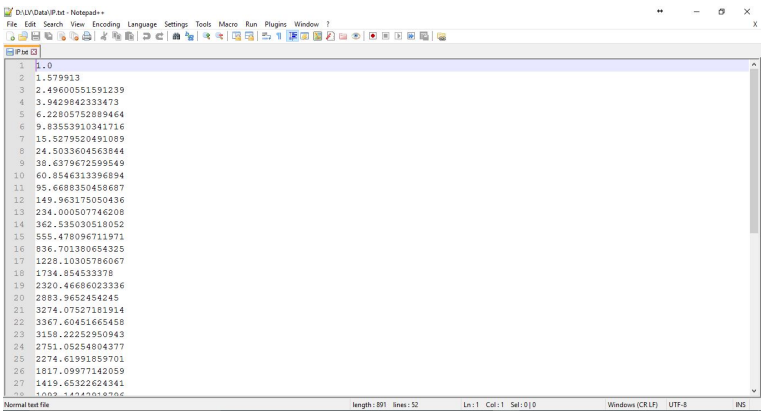


Figure 9: A snapshot of time series data format.

After filling all the variable fields, the variable can be added to the variables list by clicking on the add button above the variables list as shown in Figure 11. A notification message will be appeared to the user before adding if one of the following violations exist:

- Variable name fields are empty.
- Variable name fields contain spaces.
- Variable name fields contain only numbers or start with number.
- A time series data is not selected for output or input variable types.
- A duplication in the variable names exist.

Editing Variable The user can edit an existing variable from the variables list by clicking on one of them to load its information in the variable fields. After changing the

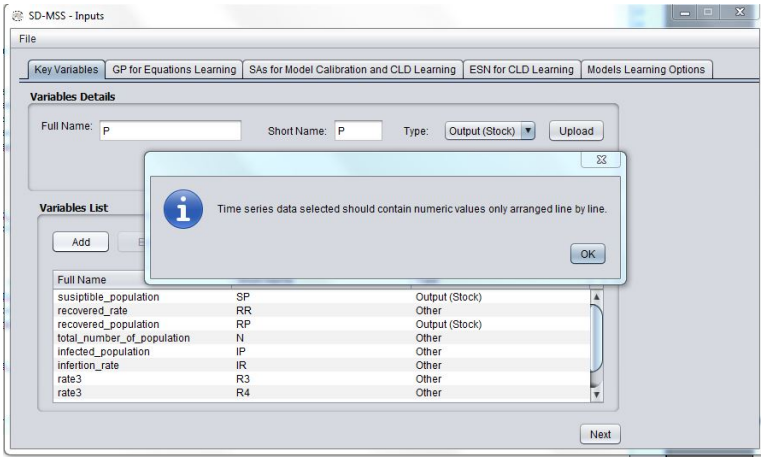


Figure 10: A snapshot of the message showed when invalid time series data format is uploaded.

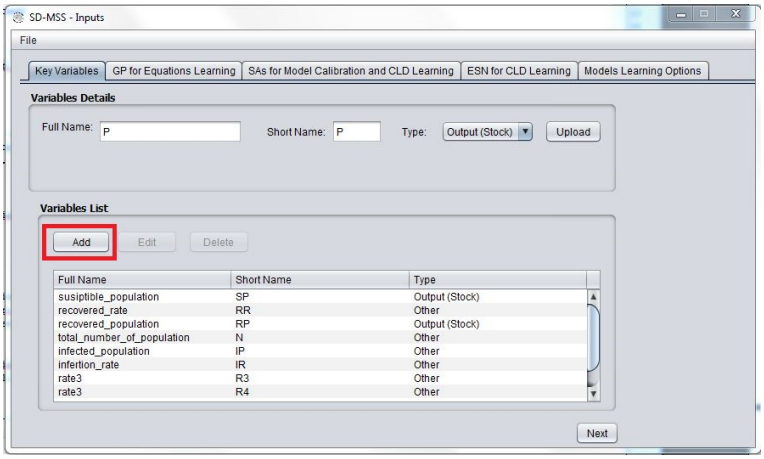


Figure 11: A snapshot of a successfully added variable to the variables list.

variable fields values with the new ones, the edit button is clicked to save and reflect these changes in the variables list as shown in Figure 12. The same constraints when adding a variable are applied for editing too.

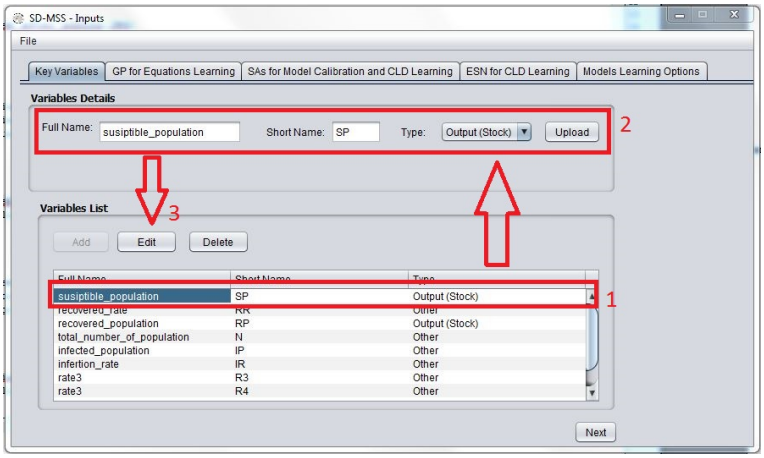


Figure 12: A snapshot of a selected variable from the variables list for editing.

Deleting Variable To delete a variable from the variables list, the user click on this variable and its related information will be loaded in the variable fields. Next, the user clicks on the delete button to remove this variable from the list as shown in Figure 13.

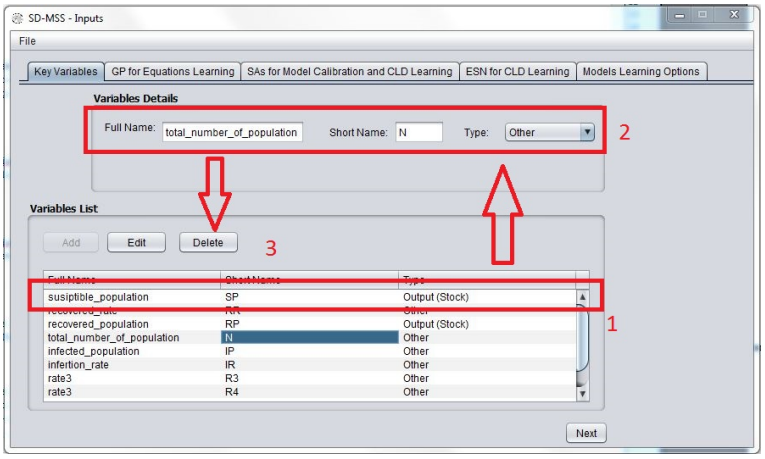


Figure 13: A snapshot of selecting a variable from the variables list for deletion.

Save Variables List After adding all system variables with their necessary information, the user can save this list by clicking on next button as shown in Figure 14. Before saving the variables list the following conditions should be satisfied:

- Variables list should not be empty.
- Variables list should contain at least one output variable and one variable of type other.
- Variables list should contain at least two variables.

Once all these constraints are satisfied the variables list is saved successfully (Figure 14), otherwise a notification message will be shown for the user. After saving the variable list by clicking on the next button, the next tab of GP parameters will be activated.

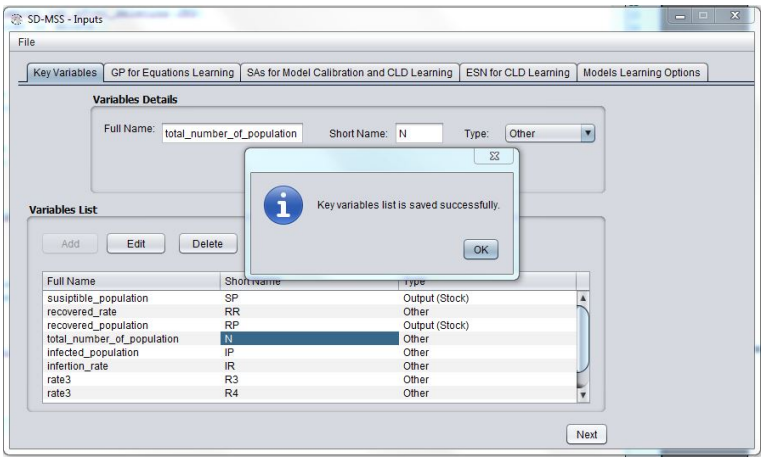


Figure 14: A snapshot of the shown message of successful saving of variables list.

4.3.2.2 GP Algorithm Tab

In this tab, the controlling parameters for the GP algorithm applied to learn system equations are viewed. As shown in Figure 6b, all the parameters have a default values and the end user could accept or edit these values. To save these parameters values the user should click the next button as shown in Figure 15.

To save the GP parameters successfully, all the fields should have numerical values and cannot be empty. In case of violating any of these conditions a proper notification

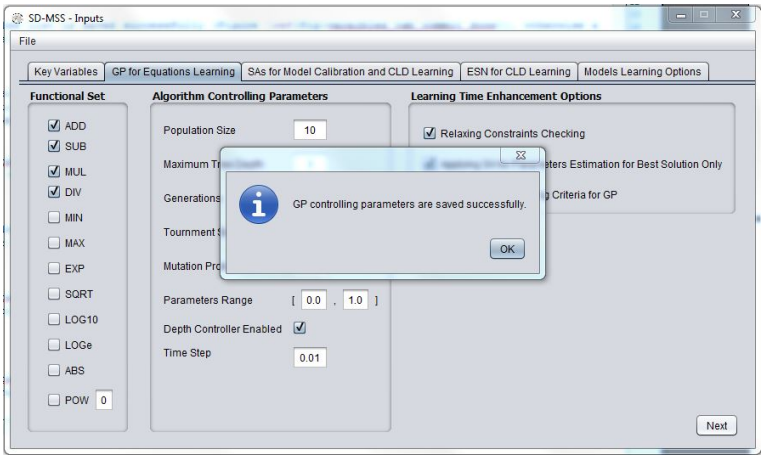


Figure 15: A snapshot of the shown message of successful saving for the GP tab parameter values.

message will be appeared. Once the GP parameters list is saved, the next tab of SA parameters is activated. In addition to the standard GP controlling parameters, we added three options under the *Learning Time Enhancement Options* panel in the right part of the tab as shown in Figure 16. These three options provide the user with the ability to reduce the learning time process in case the user want to generate learned models in short time span. If these options are disabled, the learning process will take more time, but the learned models could be more accurate. The first option relaxes the checking for the constraints when creating new individuals either at the creation of the initial population or after applying the mutation operation. The second option applies SA algorithm that is used for parameters estimation to estimate the parameters for the best found solution at the end of the GP evolution process. If this option is disabled, the SA algorithm will be applied with every single solution at each generation. The third option is applying an early stopping criteria for the GP algorithm instead of running the evolution process until reaching the maximum number of generations. The early stopping criteria checks for the fitness values for the last 10 generations, if there is no change in the fitness, then the evolution process will stop.

4.3.2.3 SA Algorithm Tab In this tab, the controlling parameters for SA algorithms applied for model calibration and CLD structure learning are identified. Each SA algorithm has its own identified settings fields as shown in Figure 6c.

All the parameters have a default values and the end user can accept or edit these values. The next button is then clicked to save these parameters values and activate the next tab as shown in Figure 17.

To save the SA parameters successfully, all the fields should have a value and this value should be numerical. In case of violating any of these constraints a proper notification message will appear.

4.3.2.4 ESN Algorithm Tab In this tab, the controlling parameters for ESN algorithms applied for CLD structure learning are identified as shown in Figure 6d.

All the parameters have a default values and the end user can accept or edit these values. The next button is then clicked to save these parameters values and activate the next tab as shown in Figure 18.

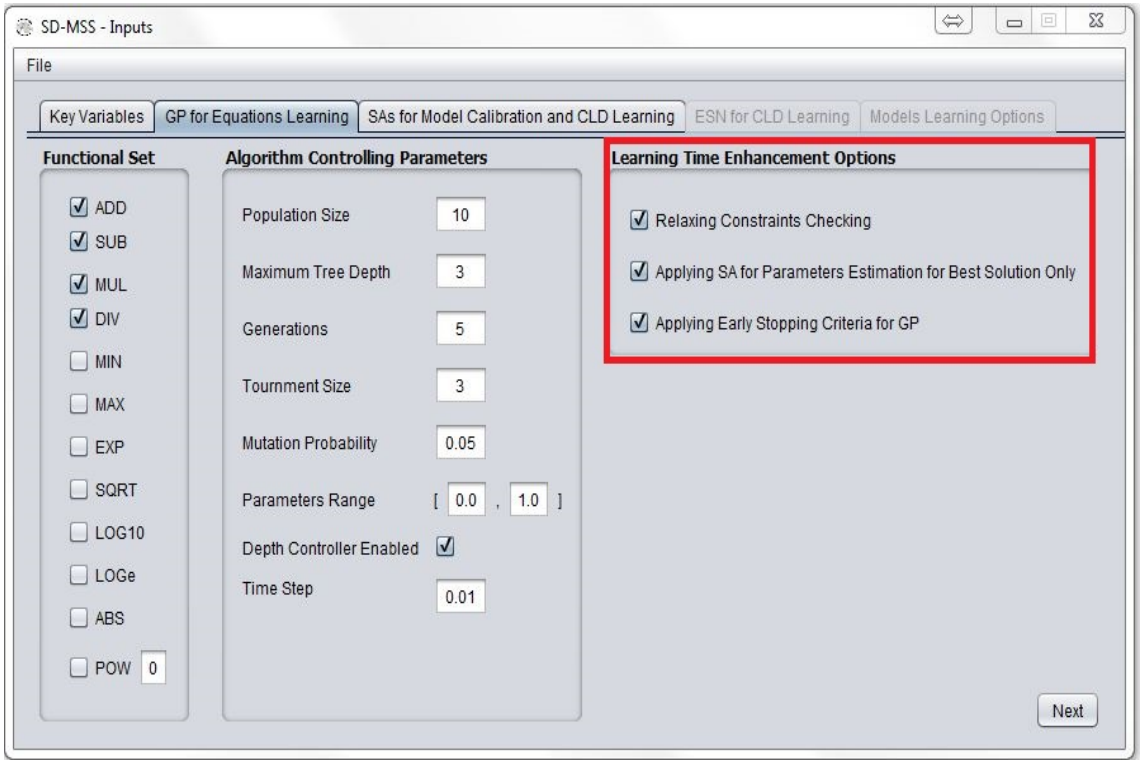


Figure 16: A snapshot of the learning time enhancement options panel in the GP tab.

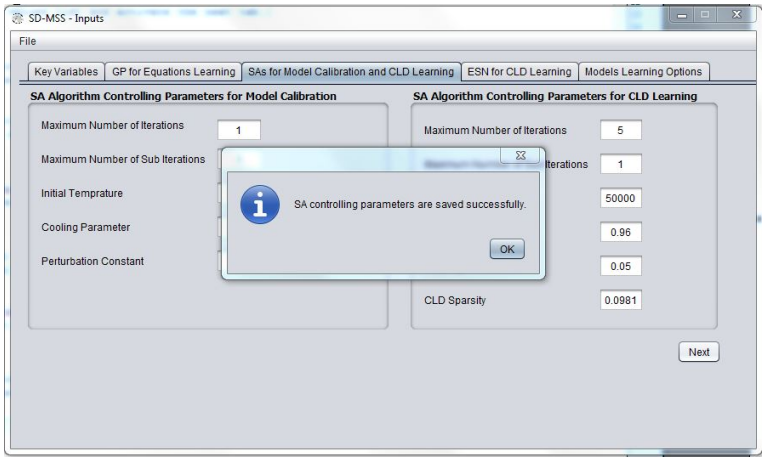


Figure 17: A snapshot of the showed message of successful saving for the SA tab parameter values.

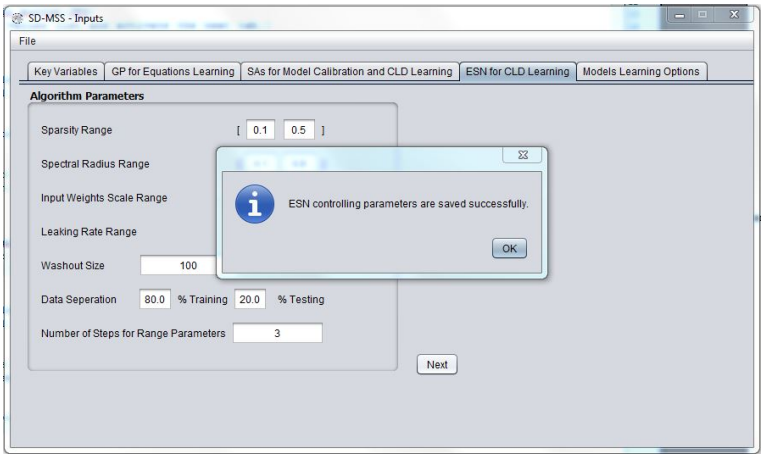


Figure 18: A snapshot of the showed message of successful saving for the ESN tab parameter values.

To save the ESN parameters successfully, all the fields should have a value and this value should be numerical. In case of violating any of these constraints a proper notification message will appear.

4.3.2.5 Learning Modes Options Tab In this tab, the user can select to either learn CLDs, system equations and estimate their parameters given a manually created CLD structure or learning CLD, equations and estimate the parameters and identifying the variables types in SFD terminologies as shown in Figure 19.

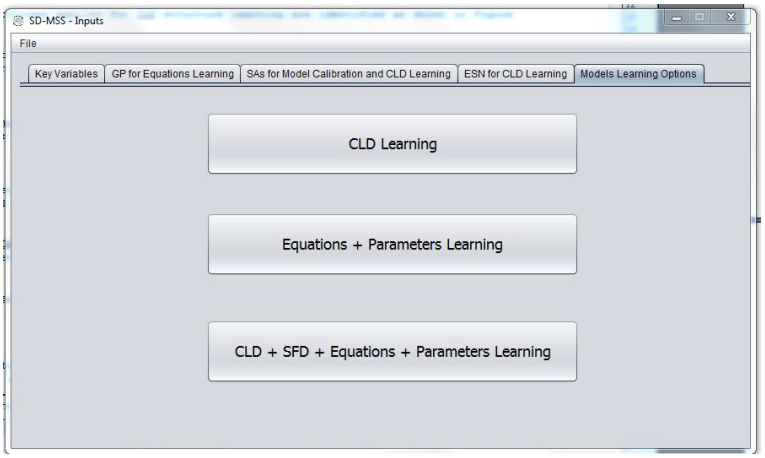


Figure 19: A snapshot of learning modes options tab.

4.3.3 CLD Learning Screen

In this screen, the user can run the ESN method for learning CLDs based on the information provided through the inputs screen. As shown in Figure 20, there are two fields, the runs and best models. The end-user can identify how many best models to be shown. These two fields cannot be empty or have any non-numerical values. In case of violating these conditions, a proper notification message will appear. After identifying these fields, the start button is clicked to start the learning process. Figure 21 shows an example of the best 10 learned models.

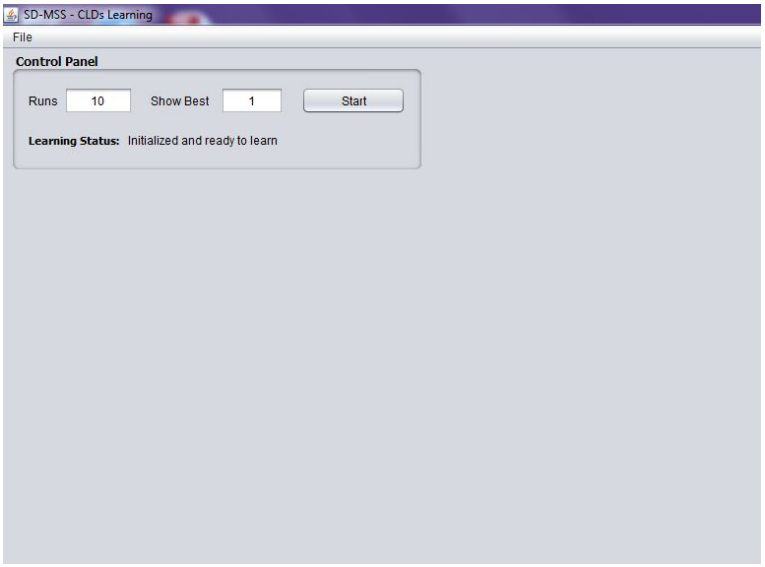


Figure 20: A snapshot of CLD learning screen before start the learning process.

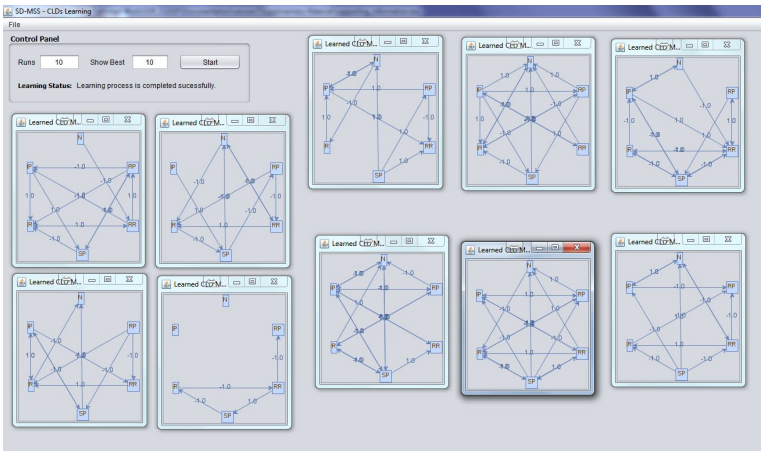


Figure 21: A snapshot of the best 10 learned CLD structures in the CLD learning screen.

The user can return back from this window to either the inputs or main windows using the file menu items same as represented in the inputs window.

4.3.4 Equations Learning and Parameters Estimation Given Manually Created CLD Screen

In this screen, the user can run the GP ensemble algorithm for equations learning and model calibration based on the provided information through the inputs screen. As shown in Figure 22, this screen consists of three parts as follows: control panel, variables relations identification and the panel that will show the learned equations.

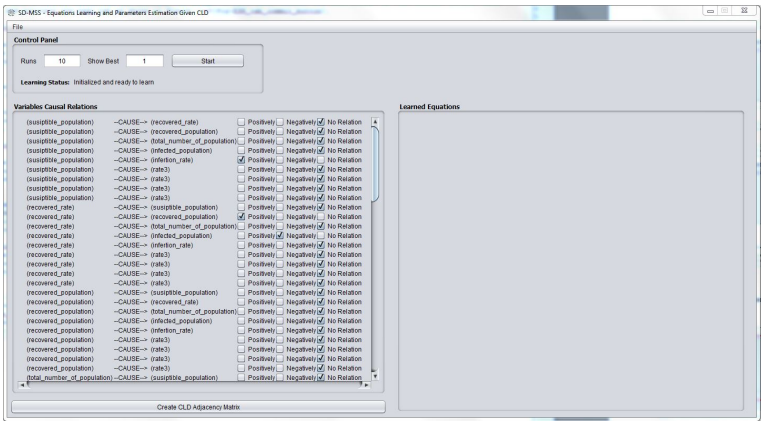


Figure 22: A snapshot of equations learning given manually created CLD screen.

The first step is to identify the causal relations among the system variables to create the target CLD. For each causal relation between two variables, three check boxes are provided: positive, negative or no relation as shown in Figure 23. Positive, negative, no-relation check boxes mean positive, negative of no causal relations, respectively.

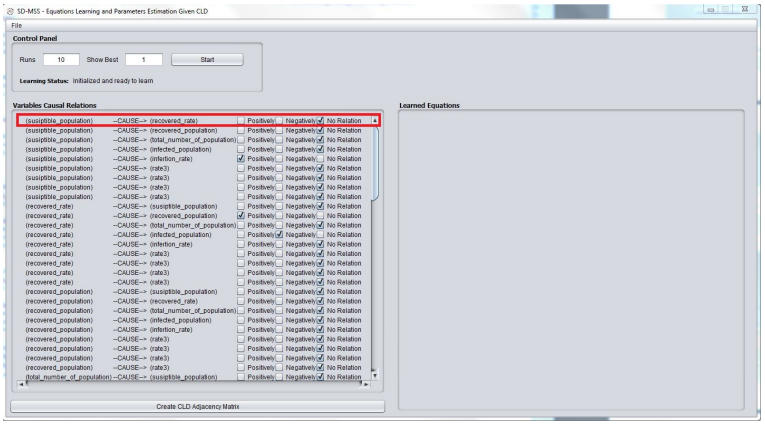


Figure 23: A snapshot of different types of relations check boxes for all pair of variables in the equations learning screen.

Once all these relations are identified, the user clicks the button of creating the CLD (Figure 24) to create the corresponding adjacency matrix given the identified relations.

For all variable pairs, if the type of relation is not identified for any pair, a notification message will be shown to the user. In addition, for any pair of variables, if more than one check box is selected a notification message will be shown too.

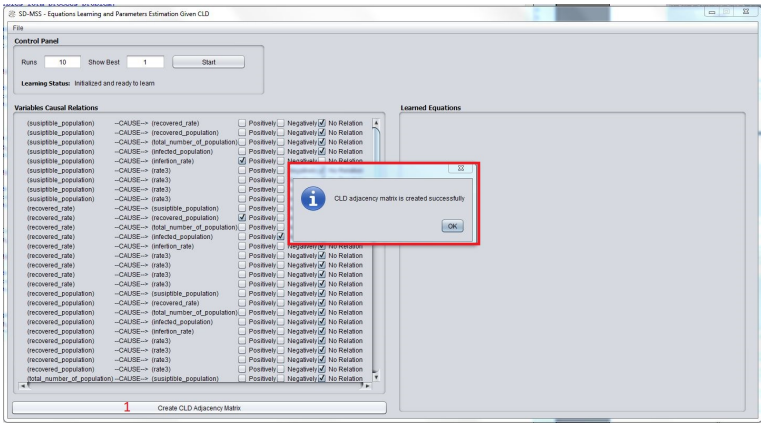


Figure 24: A snapshot of the showed message of successful creation of the CLD adjacency matrix in equations learning screen.

Once the adjacency matrix for the target CLD structure is created, the next step is to click the start button after identifying the number of runs for the algorithm and how many best models the end-user want to show as shown in Figure 25. The same constrains discussed in the previous screen of CLD learning is applied on these fields too to be non-empty and have numerical values.

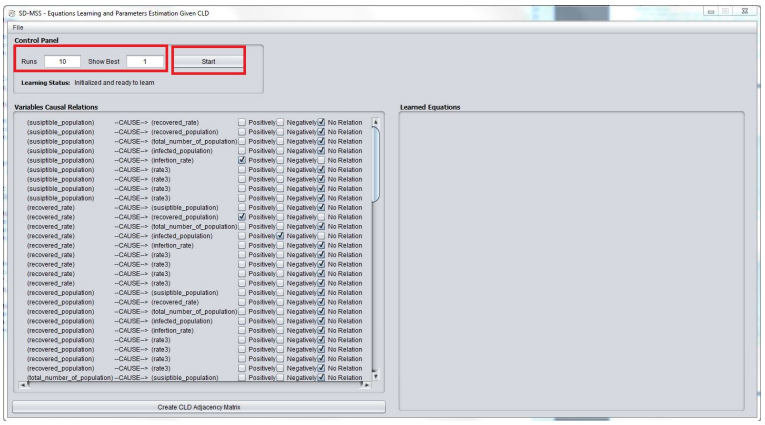


Figure 25: Start button to start the learning process by identifying number of runs and top best models to be shown.

After that, the learning process starts and once its finished the learned equations are shown in the equations panel as shown in Figure 26.

The user can return back from this screen to either the inputs or main screens using the file menu items.

4.3.5 CLD, SFD and Equations Learning Screen

In this screen, the user can run the integrated algorithms of GP ensemble and SA to learn the system CLD structure and equations, calibrate the model and identify variables types in SFD terminologies. By entering the value for runs and best models fields, the end-user can start the learning process by clicking on start button as shown in Figure 27.

After the learning process is finished, the best learned models are viewed by showing the learned equations in the left side of the screen, the variables types in the right side

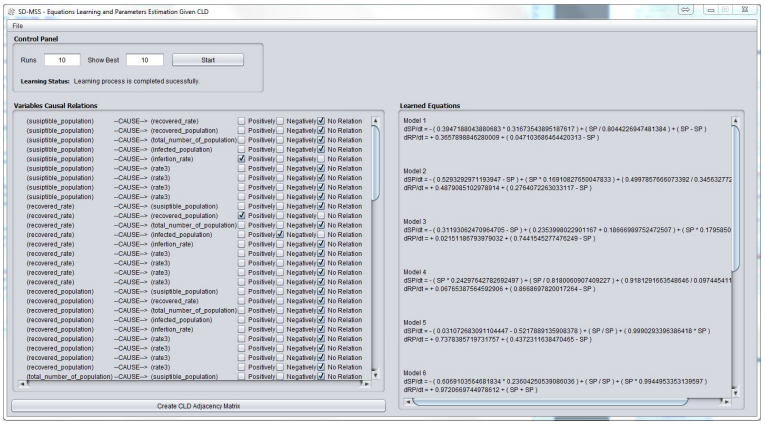


Figure 26: A snapshot of the best 10 learned equations learning in the equations learning screen.

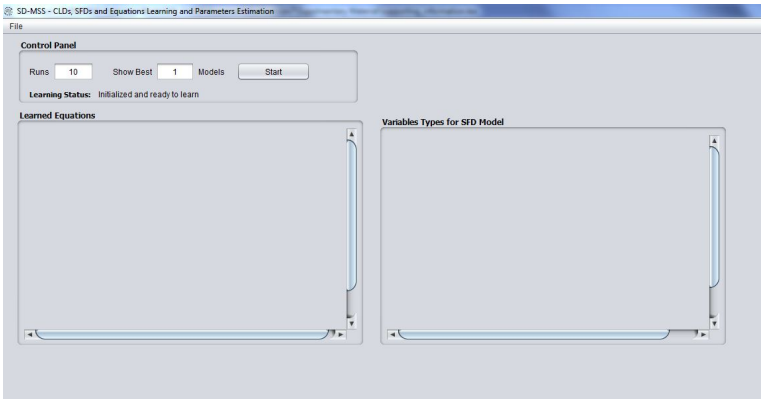


Figure 27: A snapshot of learning CLD, SFD and equations screen.

of the screen and the learned CLD structures are in a separate generated sub-screens as shown in Figure 28).

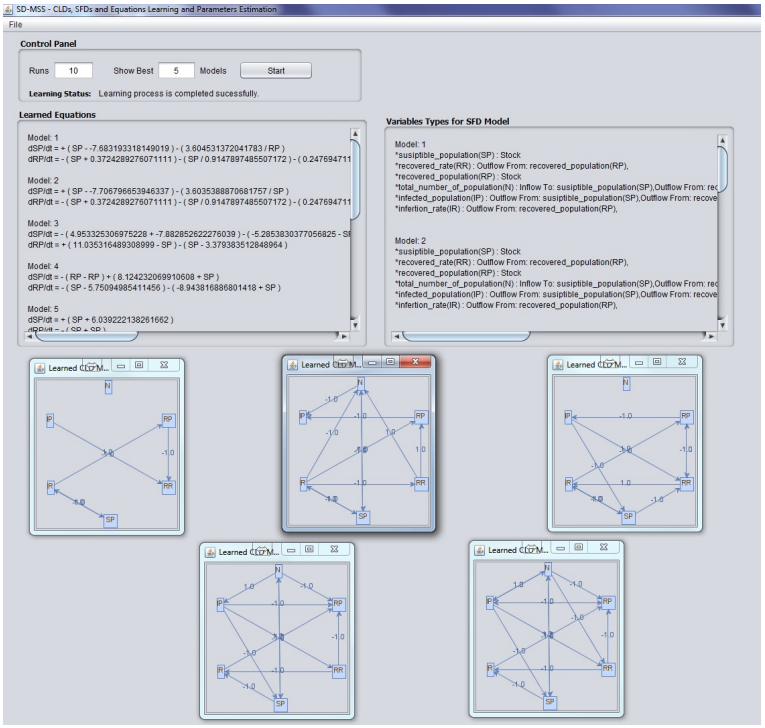


Figure 28: A snapshot of the best 10 learned models by showing the learned equations and parameters, variables types in SFD terminologies and CLD structures.

The user can return back from this screen to either the inputs or main screens using the file menu items.