
SimulAI

Release 0.0.1

Perez Colo Ivo, Pirozzo Bernardo Manuel, Saavedra Sueldo Caroli

Aug 11, 2023

CONTENTS:

1	Requirements	3
2	Installation	5
3	API Module	7
3.1	Module <code>simulai.sim</code>	7
3.2	Module <code>simulai.interface</code>	11
4	License	15
5	Indices and tables	17
	Python Module Index	19
	Index	21



SimulAI is a library that, through the use of Artificial Intelligence (AI) techniques, allows optimization of variables derived from the Simulation of Flexible Manufacturing Systems (FMS).

Given that decision-making in organizations is a complex task, due to the unpredictable dynamic nature and the large number of variables that intervene in the normal functioning of an organization, Simulai is presented with the objective, through the use of digital twins , improve the decision-making process in organizations, making it more efficient and secure.

Development

Initially the work will focus on the integration of Tecnomatix Plant Simulation software with diverse methods of optimization, which will be submitted to a process of improvement throughout the Doctorate stage in development.

In the future, it is expected to achieve a library of greater accessibility, allowing the application in various simulation software, as required by the user.

OS Windows, Tecnomatix Plant Simulation version 15.0 and pywin32 library are currently required for its operation.

Authors

- **Perez Colo Ivo** <https://github.com/IvoPerezColo>
- **Pirozzo Bernardo Manuel** - <https://github.com/BerMPirozzo>
- **Saavedra Sueldo Carolina** <https://github.com/carosaav>

Repository

<https://github.com/carosaav/SimulAI>



REQUIREMENTS

You need:

- **Python 3.7 or later**
- **OS Windows**
- **Tecnomatix Plant Simulation version 15.0**



INSTALLATION

The easiest way to install is using pip:

```
$ pip install simulai
```

This will install the latest stable version on PIPy.

If you want to use the latest development version from github, unpack or clone the [repo](#) on your local machine, change the directory to where setup.py is, and install using setuptools:

```
$ python setup.py install
```

or pip:

```
$ pip install -e .
```


API MODULE

SimulAI has the development of two automatic learning techniques:

- **Q-Learning**: is an off-policy TD control policy. It doesn't follow a policy to find the next action but rather chooses the action in a greedy fashion.
- **SARSA**: is an on-policy TD control method. Policy maps the action to be taken at each state. .

3.1 Module `simulai.sim`

Plant simulation with autonomous decision system.

class `simulai.sim.DiscreteVariable`(*name, lower_limit, upper_limit, step, path*)

Bases: object

Initialize the input Tecnomatix Plant Simulation Variables.

These variables will be used in the AI method. Up to 4 discrete variables are allowed in the problem which can form up to 625 possible states in the algorithm. For example, if 4 variables are chosen, each of them can take 5 possible values and states formed will be $S = (Var1, Var2, Var3, Var4)$.

Parameters

- **name** (*str*) – Name of the Variable.
- **lower_limit** (*positive int*) – Lower limit of the Variable. Should be a positive integer.
- **upper_limit** (*positive int*) – Upper limit of the Variable. Should be a positive integer.
- **step** (*positive int*) – Step of the Variable. Should be a positive integer.
- **path** (*str*) – Path of the Variable in Tecnomatix Plant Simulation.

class `simulai.sim.OutcomeVariable`(*name, path, column, num_rows*)

Bases: object

Initialize the output Tecnomatix Plant Simulation Variables.

These variables will be used in the AI method and must be stored in a Data Table. The chosen column from which to extract the results and the number of rows it has must be indicated.

Parameters

- **name** (*str*) – Name of the Variable.
- **path** (*str*) – Path of the Variable in Tecnomatix Plant Simulation.

- **column** (*positive int*) – Column of the table where the result is stored. Should be a positive integer.
- **num_rows** (*positive int*) – Number of rows in the results table. Should be a positive integer.

class `simulai.sim.Plant`(*method*)

Bases: `object`

Metaclass to generate various simulated manufacturing plants.

Parameters

method (*str*) – Name of the chosen AI method.

connection()

Connect function.

abstract `get_file_name_plant`()

Name of the given plant file.

abstract `process_simulation`()

Simulate in Tecnomatix.

abstract `update`(*data*)

Update.

Parameters

data (*int*) – Simulation data.

class `simulai.sim.BasePlant`(*method*, *v_i*, *v_o*, *filename*, *modelname*='Model')

Bases: `Plant`

A particularly adaptable plant.

Parameters

- **method** (*str*) – Name of the chosen AI method.
- **v_i** (*list*) – List of chosen input variables.
- **v_o** (*list*) – List of chosen output variables.
- **filename** (*str*) – Tecnomatix Plant Simulation complete file name (.spp)
- **modelname** (*str*) – Model frame name of the file, Default value="Model".

get_file_name_plant()

Get the name of the plant file.

Returns

filename – Name of the file.

Return type

`str`

update(*data*)

Update.

Parameters

data (*int*) – Simulation data.

Returns

r – Reward value.

Return type

float

process_simulation()

Process simulation.

class simulai.sim.AutonomousDecisionSystem

Bases: object

Autonomous decision system class.

register(who)

Subscribe registration.

Parameters**who** (*str*) – Node to subscribe.**abstract process()**

Process.

class simulai.sim.Qlearning(*v_i*, *episodes_max*, *steps_max*, *alfa*=0.1, *gamma*=0.9, *epsilon*=0.1, *s*=_Nothing.NOTHING, *a*=_Nothing.NOTHING, *seed*=None)Bases: [*AutonomousDecisionSystem*](#)

Implementation of the artificial intelligence method Q-Learning.

Whose purpose is to obtain the optimal parameters from the trial and error method, in which it is penalized if the goal is not reached and is rewarded if it is reached, requiring for this a number of episodes. The Q table has a maximum of 625 rows, that is, up to 625 states are supported. These states are made up of 1 to 4 variables of the Tecnomatix Plant Simulation. Actions also depend on the chosen variables and their steps. The reward function depends on the results defined in the respective plant class.

Parameters

- **v_i** (*list*) – List of chosen input variables.
- **episodes_max** (*positive int*) – Total number of episodes to run. Should be a positive integer.
- **steps_max** (*positive int*) – Total number of steps in each episode. Should be a positive integer.
- **alfa** (*float*) – Reinforcement learning hyperparameter, learning rate, varies from 0 to 1. Default value= 0.10
- **gamma** (*float*) – Reinforcement learning hyperparameter, discount factor, varies from 0 to 1. Default value= 0.90
- **epsilon** (*float*) – Reinforcement learning hyperparameter, probability for the epsilon-greedy action selection, varies from 0 to 1. Default value= 0.10
- **seed** (*int*) – Seed value for the seed() method. Default value=None.

arrays()

Arrays for states and actions.

ini_saq()

Initialize states, actions and Q table.

choose_action(row)

Choose the action to follow.

Parameters**row** (*int*) – Number of rows.**Returns****i** – Selected row.**Return type****int****process()**

Learning algorithms.

Returns**r_episode** – Episode reward**Return type****float**

```
class simulai.sim.Sarsa(v_i, episodes_max, steps_max, alfa=0.1, gamma=0.9, epsilon=0.1,
                        s=_Nothing.NOTHING, a=_Nothing.NOTHING, seed=None)
```

Bases: [Qlearning](#)

Implementation of the artificial intelligence method Sarsa.

Whose purpose is to obtain the optimal parameters from the trial and error method, in which it is penalized if the goal is not reached and is rewarded if it is reached, requiring for this a number of episodes. The Q table has a maximum of 625 rows, that is, up to 625 states are supported. These states are made up of 1 to 4 variables of the Tecnomatix Plant Simulation. Actions also depend on the chosen variables and their steps. The reward function depends on the results defined in the respective plant class.

Parameters

- **v_i** (*list*) – List of chosen input variables.
- **episodes_max** (*positive int*) – Total number of episodes to run. Should be a positive integer.
- **steps_max** (*positive int*) – Total number of steps in each episode. Should be a positive integer.
- **alfa** (*float*) – Reinforcement learning hyperparameter, learning rate, varies from 0 to 1. Default value= 0.10
- **gamma** (*float*) – Reinforcement learning hyperparameter, discount factor, varies from 0 to 1. Default value= 0.90
- **epsilon** (*float*) – Reinforcement learning hyperparameter, probability for the epsilon-greedy action selection, varies from 0 to 1. Default value= 0.10
- **seed** (*int*) – Seed value for the seed() method. Default value=None.

process()

Learning algorithm.

Returns**r_episode** – Episode reward**Return type****float**

3.2 Module `simulai.interface`

Class makes the connection with Tecnomatix Plant Simulation.

exception `simulai.interface.ConnectionError`

Bases: `Exception`

Connection failed exception.

exception `simulai.interface.ModelNotFoundError`

Bases: `FileNotFoundError`

Custom error for Not found Model.

`simulai.interface.check_connection(method)`

Check the connection status, returning an error.

Parameters

method (*str*) – Name of the method.

Return type

A message indicating failure.

class `simulai.interface.CommunicationInterface(model_name, is_connected=False, plant_simulation="")`

Bases: `object`

Definition of the function of communication.

Parameters

model_name (*str*) – Name of the Tecnomatix Plant Simulation file.

is_connected

Connection status

Type

`bool`

plant_simulation

Attribute for the return of the connection object.

Type

`str`

get_path_file_model()

Return the complete file path.

Returns

file path – Path of Tecnomatix Plant Simulation file.

Return type

`str`

connection()

Return the connection object.

Returns

connection status – Connection indicator.

Return type

`bool`

setvisible(*value*)

Execute the application Tecnomatix.

Parameters

value (*bool*) – User-selected value.

setvalue(*ref*, *value*)

Set the values in the simulator.

Parameters

- **ref** (*str*) – Path of the variable.
- **value** (*int*) – User-selected value.

getvalue(*ref*)

Get the values of the simulator.

Parameters

ref (*str*) – Path of the variable.

startsimulation(*ref*)

Make the simulation start.

Parameters

ref (*str*) – Path of the model.

resetsimulation(*ref*)

Make the simulation reset.

Parameters

ref (*str*) – Path of the model.

stopsimulation(*ref*)

Make the simulation stop.

Parameters

ref (*str*) – Path of the model.

closemodel()

Close the simulation model.

execute_simtalk(*ref*, *value*)

Execute the simulation programming language.

Parameters

- **ref** (*str*) – Path of the variable.
- **value** (*int*) – User-selected value.

is_simulation_running()

Check if the simulation is running.

loadmodel(*ref*, *value*)

Perform the load of the model.

Parameters

- **ref** (*str*) – Path of the file.
- **value** (*int*) – User-selected value.

newmodel()

Create a new model.

openconsole_logfile(*ref*)

Open the simulation result in the console.

Parameters

ref (*str*) – Path of the file.

quit()

Clear all result.

quit_aftertime(*value*)

Clear all result after a time.

Parameters

value (*int*) – User-selected value.

savemodel(*ref*)

Save the model result.

Parameters

ref (*str*) – Path of the file.

set_licensetype(*ref*)

Set the type of the license.

Parameters

ref (*str*) – Path of the file.

set_no_messagebox(*value*)

Delete the messages on the screen.

Parameters

value (*int*) – User-selected value.

set_pathcontext(*ref*)

Set the context.

Parameters

ref (*str*) – Path of the file.

set_suppress_start_of_3d(*value*)

Eliminate the start of 3D model.

Parameters

value (*int*) – User-selected value.

set_trustmodels(*value*)

Set the real model.

Parameters

value (*int*) – User-selected value.

transfermodel(*value*)

Transfer the model.

Parameters

value (*int*) – User-selected value.

LICENSE



Copyright (c) 2020, Perez Colo Ivo, Pirozzo Bernardo Manuel, Saavedra Sueldo Carolina

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

INDICES AND TABLES

- `genindex`
- `search`

PYTHON MODULE INDEX

S

`simulai.interface`, [11](#)

`simulai.sim`, [7](#)

INDEX

A

arrays() (*simulai.sim.Qlearning method*), 9
AutonomousDecisionSystem (*class in simulai.sim*), 9

B

BasePlant (*class in simulai.sim*), 8

C

check_connection() (*in module simulai.interface*), 11
choose_action() (*simulai.sim.Qlearning method*), 9
closemodel() (*simulai.interface.CommunicationInterface method*), 12
CommunicationInterface (*class in simulai.interface*), 11
connection() (*simulai.interface.CommunicationInterface method*), 11
connection() (*simulai.sim.Plant method*), 8
ConnectionError, 11

D

DiscreteVariable (*class in simulai.sim*), 7

E

execute_simtalk() (*simulai.interface.CommunicationInterface method*), 12

G

get_file_name_plant() (*simulai.sim.BasePlant method*), 8
get_file_name_plant() (*simulai.sim.Plant method*), 8
get_path_file_model() (*simulai.interface.CommunicationInterface method*), 11
getvalue() (*simulai.interface.CommunicationInterface method*), 12

I

ini_saq() (*simulai.sim.Qlearning method*), 9
is_connected (*simulai.interface.CommunicationInterface attribute*), 11

is_simulation_running() (*simulai.interface.CommunicationInterface method*), 12

L

loadmodel() (*simulai.interface.CommunicationInterface method*), 12

M

ModelNotFoundError, 11
module
 simulai.interface, 11
 simulai.sim, 7

N

newmodel() (*simulai.interface.CommunicationInterface method*), 12

O

openconsole_logfile() (*simulai.interface.CommunicationInterface method*), 13
OutcomeVariable (*class in simulai.sim*), 7

P

Plant (*class in simulai.sim*), 8
plant_simulation (*simulai.interface.CommunicationInterface attribute*), 11
process() (*simulai.sim.AutonomousDecisionSystem method*), 9
process() (*simulai.sim.Qlearning method*), 10
process() (*simulai.sim.Sarsa method*), 10
process_simulation() (*simulai.sim.BasePlant method*), 9
process_simulation() (*simulai.sim.Plant method*), 8

Q

Qlearning (*class in simulai.sim*), 9
quit() (*simulai.interface.CommunicationInterface method*), 13

`quit_aftertime()` (*simulai.interface.CommunicationInterface method*),
13

R

`register()` (*simulai.sim.AutonomousDecisionSystem method*), 9
`resetsimulation()` (*simulai.interface.CommunicationInterface method*),
12

S

`Sarsa` (*class in simulai.sim*), 10
`savemodel()` (*simulai.interface.CommunicationInterface method*), 13
`set_licensetype()` (*simulai.interface.CommunicationInterface method*),
13
`set_no_messagebox()` (*simulai.interface.CommunicationInterface method*),
13
`set_pathcontext()` (*simulai.interface.CommunicationInterface method*),
13
`set_suppress_start_of_3d()` (*simulai.interface.CommunicationInterface method*),
13
`set_trustmodels()` (*simulai.interface.CommunicationInterface method*),
13
`setvalue()` (*simulai.interface.CommunicationInterface method*), 12
`setvisible()` (*simulai.interface.CommunicationInterface method*), 11
`simulai.interface`
 module, 11
`simulai.sim`
 module, 7
`startsimulation()` (*simulai.interface.CommunicationInterface method*),
12
`stopsimulation()` (*simulai.interface.CommunicationInterface method*),
12

T

`transfermodel()` (*simulai.interface.CommunicationInterface method*),
13

U

`update()` (*simulai.sim.BasePlant method*), 8
`update()` (*simulai.sim.Plant method*), 8