emergylib

Release 1.0

Chams Lahlou, Olivier Le Corre and Laurent Truffet

Feb 24, 2021

CONTENTS:

CHAPTER

ONE

EMERGYLIB

A python library for computing emergy.

1.1 Overview

bla bla

1.2 Installation

Install Python 3 if necessary.

Download `emergy package`.

Unzip the package

1.3 Documentation

1.3.1 emergylib package

emergylib.graph module

This module implements class *Graph*.

class emergylib.graph.**Graph**
Bases: object

This class is used to define an emergy system as a directed graph. A node represents a physical element of the system, such as a source, a product, a split, a coproduct or a tank. An arc represents a physical link bewteen two elements.

add_arc(*node_label*, *suc_label*, *weight=1.0*, *is_fast=True*, *mass=1.0*)
Add an arc to the graph.

Parameters

- **node_label** (*string or integer*) – label of the head node.
- **suc_label** (*string or integer*) – label of the tail node.
- **weight** (*double*) – percentage of the incoming flow which follows the arc.
- **is_fast** (*boolean*) – True if it takes at most one time step to pass through.

- **mass** (*double*) – length * density * cross_section of the link (in kg).

add_node(*node_label*, *node_type*, **args*)
Add a node to the graph.

Parameters

- **node_label** (*string or integer*) – label of the node.
- **node_type** (*'source', 'product', 'split', 'coproduct', or 'tank'*) – type of the node.
- ***args** (*integer*) – solar equivalent joule if node_type is 'source'.

load(*file_name*)
Load the characteristics of the graph. For the format of the file see function `save_system()`.

Parameters **file_name** (*string*) – name of the load file

normalize()
Sort nodes and arcs in lexicographic order.

reachable_products()
Compute the list of products reachable from each source. A product is reachable from a source if there is a path from the source to the product in the graph.

Returns a dict of {str:list of int}, where each key is a source label and the associated value is the list of reachable products from the source.

save(*file_name*)
Save the characteristics of the graph.

Parameters **file_name** (*string*) – name of the save file

Each line of the file contains the characteristics of either an element or a link between two elements.

For a source element a line is of the form

'SOURCE' LABEL PARAM1

where

- LABEL is the label of the source;
- PARAM1 is the sej value of the source.

For other types of element a line is of the form

TYPE LABEL

where

- TYPE is either the string 'SPLIT', 'COPRODUCT', 'TANK' or 'PRODUCT';
- LABEL is the label of the element.

For a link a line is of the form:

'LINK' LABEL LABEL2 WEIGHT IS_FAST MASS

where

- LABEL is the label of the head element of the link;
- LABEL2 is the label of the tail element of the link;
- WEIGHT is the weight of the link;
- IS_FAST is True or False, for the time delay of the link;

- MASS is the mass of the link.

emergylib.state module

This module implements class *State*.

class emergylib.state.**State**(*graph*, *time_step*)
Bases: object

This class records the state of the system at current time.

Parameters

- **graph** (object of class *Graph*) – graph of the system.
- **time_step** (*integer*) – delay between two updates of the system.

update(*source_flow*, *tank_flow*, *tank_load*, *is_operational*)
Set the new state for the system at current time, i.e. assign new flow values to sources and tanks, new load values to tanks, and indicate if links are working or not.

Parameters

- **source_flow** (*dict {int:float}*) – new flow values of sources.
- **tank_flow** (*dict of {int:float}*) – new outgoing flow value of tanks.
- **tank_load** (*dict of {int:float}*) – load values of tanks.
- **is_operational** (*dict of {(int, int):bool}*) – new state of arcs.

emergylib.system module

This module implements class *System*. This class is used to model an emergy system and compute emergy and empower of products.

class emergylib.system.**System**(*time=0*, *time_step=1*, *epsilon=0.01*, *num_cesaro=5*, *max_delay=0*)
Bases: object

add_arc(*node_label*, *suc_label*, *weight=1.0*, *is_fast=True*, *mass=1.0*)
Add an arc between two nodes to the system.

Parameters

- **node_label** (*string or integer*) – label of the head node.
- **suc_label** (*string or integer*) – label of the tail node.
- **weight** (*double*) – percentage of the incoming flow which follows the arc.
- **is_fast** (*boolean*) – True if the link is working. False otherwise
- **mass** (*double*) – length * density * cross_section of the link (in kg).

add_coproduct(*node_label*)
Add a coproduct node to the system.

Parameters **node_label** (*string*) – label of the coproduct.

add_product(*node_label*)
Add a product node to the system.

Parameters **node_label** (*string*) – label of the product.

add_source(*node_label*, *sej*)
Add a source node to the system.

Parameters

- **node_label** (*string*) – label of the source.
- **sej** (*integer*) – unitary emergy value in solar equivalent joule.

add_split(*node_label*)
Add a split node to the system.

Parameters **node_label** (*string*) – label of the split.

add_tank(*node_label*)
Add a tank node to the system.

Parameters **node_label** (*string*) – label of the tank.

calibrate(*display=False*)
Set the value of self.__max_delay. By default, it is equal to 5 times the delay of flowing one input. See funtion *run_static()*.

create()
Set initial values and create first state of the system.

load(*file_name*)
Load the characteristics of the graph and create the associated system. For the format of the file see function `save_system()`.

run_scenario(*input_name*, *output_name=None*, *display=False*)

run_static(*display=False*)

save(*file_name*)
Save the characteristics of the graph.

update(*source_flow*, *tank_load*, *tank_flow*, *is_operational*, *calibration=False*)
Return the new emergy and empower values of the products, after doing the following:

1. Set new flow values, load values, and state of the links.
2. Add input nodes in the unfolded graph.
3. Add new nodes to the unfolded graph.
4. Compute emergy and empower of products.
5. remove non active nodes of the unfolded graph.
6. Increment current time.

Parameters

- **source_flow** (*dictionary of integer:double*) – new flow values of sources.
- **tank_flow** (*dictionary of integer:double*) – new outgoing flow value of tanks.
- **tank_load** (*dictionary of integer:double*) – load values of tanks.
- **is_operational** (*dictionary of (integer, integer):boolean*) – new state of arcs.

Returns new emergy and flow values of products.

Return type two dictionaries of integer:double

write_sim_header(*output_name*)

emergylib.ugraph module

This module implements class `UGraph`. TITI

emergylib.simfile module

This module implements class *SimFile*.

class emergylib.simfile.**SimFile**(*graph*, *input_name*, *output_name=None*)
Bases: object

This class is used to simulate the behaviour of the system over time. An input file is used to read input values. Emergy and empower results can be saved in an output file. The format of a simulation input file is defined as follows:

- **First line contains the label of columns in the following order:**
  1. Source labels for the input flows
  2. Tank labels for the outgoing flows
  3. Tank labels for the loads
  4. Arc labels for the link status. An arc label is of the form 'head:tail'
- Next lines contain the values to read. Each line corresponds to an update.

The format of a simulation output file is defined as follows:

- **First line contains the label of columns in the following order:**
  1. Product labels for emergy values
  2. Product labels for empower values
- Next lines contain the values saved. Each line corresponds to an update.

Parameters

- **graph** (object of class *Graph*) – name of the input file.
- **input_name** (*string*) – name of the input file.
- **output_name** (*string*) – name of the output file.

close()
Close input and output files.

read_line(*s_flows*, *t_flows*, *t_loads*, *is_operational*)
Read the current line of the input file to get source flows, tank flows and loads, and link status for the current time.

Parameters

- **source_flow**(*dict {int:float}*) – flow values of sources.
- **tank_flow**(*dict of {int:float}*) – outgoing flow value of tanks.
- **tank_load**(*dict of {int:float}*) – load values of tanks.

- **is_operational** (*dict of {(int, int):bool}*) – state of arcs.

write_line(*emergy*, *empower*)

Write emergy and empower values of products for the current time. A new line is added to the output file.

Parameters

- **emergy** (*dict of {int:float}*) – emergy values of products.
- **empower** (*dict of {int:float}*) – empower values of products.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX