GEA

Release v.0.1

Capgemini Engineering - Hybrid Intelligence

Mar 15, 2022

SCHEDULER CONTENTS:

The scheduler optimization module is formulated as a Mixed-Integer Programming problem (MIP) using the framework ORTools and Python as programming language.

$$min_{x \in \mathbb{R}, y \in (0,1)} \; f(x, y)$$

$$st: \; g(x, y) <= 0$$

The module evaluates whether a certain plant configuration is able to satisfy a certain product demand within a given time frame (both also given as an input parameter) according to some solution criterion to be clarified below.

Within the scheduler module, an initial pre-check (input data analyzer) is executed before starting the scheduling problem to identify possible data inconsistencies and will communicate the possible warnings or errors founded in the output of the scheduler.

If there exists a feasible schedule for the input plant configuration, the scheduler provides a schedule-type output, which will later be used to construct one Gantt chart (the construction is out of this scope).

If no feasible schedule exists, then the relaxed schedule-type output is provided, and the list of the equipment involved with their utilization ratios will be provided. This will be obtained by relaxing certain constraints until a feasible problem can be reached to follow the same procedure as with bottleneck identification.

If the input data instance is not consistent (exist errors) then no schedule-type output is provided. The error message will indicate clearly the reason of failure.

The output will contain a list of possible warnings and error (data inconsistencies) messages.

This version includes:

- Batch/continuous processes.
- Already running equipment.
- Multi-product/Multi-workflows.

This version DOES NOT includes:

- CIPs.
- Product Order.
- Clusters.
- Split Workflows.

CHAPTER

ONE

SCHEDULER MODULE DOCUMENTATION

1.1 Model Data

class src.model_data.**ModelData**

Model Data

This class defines the data model of the input instance. It contains the main attributes that the scheduler requires.It also contain the different methods required to work with the attributes.

Attributes: schedule_configs: Dictionary that contain the schedule configuration:

objective: objective to consider (1:makespan, 2:just-in-time). String.

product_order: order of the products to be scheduled. List:

starting_date: Starting date to consider in the schedule. Timestamp (dd-mm-yyyy hh:mm:ss).

max_time_horizon: Max. number of days to consider in the schedule. Integer.

time_resolution: Time resolution of the schedule (1-30 min). Integer

plant_ID: Identifier for the plant configuration. It wil be used for the schedule_ID. String

equipment: Dictionary that contains the information of the equipment involved in the schedule. Each register is a equipment.

equipment_ID: Identifier for the equipment.

no_inputs: Number of inputs.

no_outputs: Number of outputs.

batch_max: Maximum batch size.

batch_min: Minimum batch size.

calendar: Equipment_calendar. Working horus. format: [24,7].

demand: Dictionary that contains the information of the product demand:

product_ID: Identifier for th product. Is a list.

due_data: Dictionary that contain the due_date for each product.

amount: Dictionaty that contain the quantity demanded for each product

workflows: Dictionary that contains the information related to recipe/workflows:

workflow_ID: Identifier for the workflow.

input_product: Input product/s. [String].

output_product: Ouput product/s. [String].

recipe: Dictionary that contains the recipes (workflow graph)

- node: Dictionary that contains:
  - node_ID
  - subprocess
  - duration_type
  - duration
- edge: Dictionary that contains:
  - node_origin_ID
  - node_destination_ID
  - subprocess_destination
  - subprocess_origin
  - product_origin
  - product_destination
  - flow_rate
  - delay
  - connection: Type of connection (SS, FF, FS, SF)

1.2 Scheduler Response

`class src.scheduler_response.SchedulerResponse`

Scheduler Response

This class defines the schedule data class of the scheduler engine. It contains the necessary information to build the schedule gantt chart.

Attributes: schedule_status: Indicate the status of the schedule (feasible, not_feasible or not_consistent)

equipment_unfeasible: It is a dictionary with all the equipment involved in the schedule with the utilization ratios (UR).

status_data_analyzer: It is a dictionary with warning and errors messages

schedule_ID: Identifier for the schedule (string)

schedule_gantt: Dictionary that contains:

- equipment_ID: Equipment involved
- workflow_ID: Workflow associated
- product_order: Product Order associated
- work_order: Work Order associated
- job_order: job_order associated

product: Product associated

subprocess: Subprocess

start: Start timestep

end: End timestep

volume: Quantity of product

1.3 Scheduler Engine

`class src.engine.SchedulerEngine(data: src.model_data.ModelData)`

Scheduler Engine

This class defines the scheduler optimization engine.

Attributes: results: Results of the pyomo execution.

name: Name of the model.

data: Model data instance.

response: Scheduler instance.

`execute()`

Execute

This method execute the engine module which comprise the following sub-methods:

- Data Analyzer: Check the integrity of the input data instance.
- Build Model: Build the engine scheduler model.
- Solve: Solve the optimziation model.
- Build Solution: Invoke the scheduler data factory to build the schedule output solution

1.4 Model Data Factory

`class src.model_data_factory.ModelDataFactory(request_path)`

Model Data Factory

This class creates an instance of the Model data class with the input of the scheduler engine.

Attributes: request_path path or json file data model data class

`static create(request_path)`

Create

This method creates an instance of the model data class with the input json file provided

Attributes: request_path: input json file

1.5 Scheduler Response Factory

class `src.scheduler_response_factory.`**`SchedulerResponseFactory`**(*model*, *data: src.model_data.ModelData*)

Scheduler Response Factory

This class creates an instance of the Schedule Response class with the output of the scheduler engine.

Attributes: response: scheduler data class.

model: Scheduler engine class.

data: model data class.

static create(*model*, *data*)

Create

This method creates an instance of the schedule response data class with the solution provided by the engine scheduler.

Attributes: data: Receive the model data class of the schedule.

1.6 Utils

`src.utils.`**`check_environment`**()

Check Environment

Check if the python environment is correctly configured

`src.utils.`**`initialize_logger`**(*name*)

Initialize Logger

Initialize the logger functionality to capture the progress of the execution

`src.utils.`**`list_to_reason`**(*self*, *exception_list*)

List to Reason

Raise an exception list

`src.utils.`**`to_dict`**(*df*, *index=None*)

To dict

CHAPTER

TWO

UNIT TESTING DOCUMENTATION

2.1 Data Test

class `test.test_data.`**`DataTest`**(*methodName='runTest'*)

Data Test

This class defines the unitary test for the input data instance of scheduler

`test_connectivity`()

Test Connectivity of the plant configuration

Test if the equipment of the plant configuration is connected to the rest of the equipment of the workflow

`test_consistency`()

Test consistency of the workflows

Test if the workflows defined are consistent

`test_data_input`()

Test Data Input

Test if some input data instances are feasible plant configuration for the scheduler engine

Input data instances:

- example_test_1.json
- example_test_2.json
- example_test_3.json

`test_feasible_time_horizon`()

Test feasible time horizon

Test if the time horizon provided is feasible to allocate the longest schedule

2.2 Engine Test

class `test.test_engine.`**`EngineTest`**(*methodName='runTest'*)

Engine Test

This class defines the unitary test for the engine scheduler

`test_engine`()

Test engine

Test scheduler engine with different input plant configurations. Check that the output schedule fulfill the expected functionalities.

Input data instances:

- example_test_1.json
- example_test_2.json
- example_test_3.json

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX