Project Structure – Multi-objective optimization

1. Build – contains compiled cython (C + python) files

2. Data – Contains data files of benchmark problems and inspired from real-life production problems
    a. Benchamark - directory for benchmark problem files
    b. Plots – Contains solution space plots for GA and TS
    c. Result_JSSP – rescheduling experiment results of JSSP
    d. Result_FJSSP – rescheduling experiment results of FJSSP
    e. *.plfow - Petri net Graph files
    f. Schedule.h5 - Obtained schedule in HDF format
    g. Job_info_x.xlsx- information of job, operations, demand (no of pieces to produce), operation sequence, required machines
    h. Machine.xlsx - information of machine id, machine processing speed, setup time, breakdown threshold, average processing time, repair duration. (Note: setup time is not considered in experiments)
    i. Sequence_dependecy_matrix – operation sequence dependency matrix (not considered)
    j. Stock.xlsx - information of stock id, stock type, stock level
    k. *.txt - text files which logs the production process performance with respect to each job

3. Example_output – Used to store schedule output from algorithm in excel format and HTML files that contains plots of algorithms performance.

4. Optimizer – Repository of implemented hybrid algorithm
    a. Genetic_alg – Contains implemented genetic algorithm files
        i. _ga_helpers.pyx – helper file for the Genetic algorithm (used cython for faster computation -> heap structure in C)
        ii. Genetic_alg.py - Contains developed Genetic algorithm code

    b. Simulated_annealing - Contains implemented simulated annealing algorithm (not used in hybrid approach)
        i. _generate_neighbor.pyx - cython file contains function to generate neighbourhood of seed solution
        ii. Simulated_annealing.py - Contains developed Simulated Annealing algorithm code

    c. Solution – Contains files to generate solutions for the algorithms, and to create schedule in time-framed format.

i. _makespan.pyx - contains code to evaluate makespan of the solution or schedule (not used)
ii. _schedule_creator.py - contains code to generate time-framed schedule in excel format
iii. factory.py - contains code to generate initial population or set of solutions based on randomess and dispatching rules. (Chromosome generation)
iv. Solution.py - contains code to decode the chromosome representation and to evaluate objective function values
v. Utility.py - contains code to introduce buffer time for a machine based on its breakdown probability

d. Tabu_search – contains implemented tabu search files
    i. _generate_neighbor.pyx - cython file contains function to generate neighbourhood of seed solution
    ii. Tabu_search.py - Contains developer tabu search code

e. Template – contains template for benchmark result plots in HTML format
f. Benchmark_plotter.py - Python code to plot algorithm performance results
g. Coordinator.py - Important file in the developed algorithm code. Gets hyperparameter and sequential ensemble information from main executioner and executes Genetic algorithm and parallel tabu search in the mentioned order.
h. Data_fjs.py - Python code to extract the inspired real-life problem's data
i. Data_normal_job_shop.py – Python code to extract data from given benchmark problem's file
j. Data.py - Acts as super class for data extraction files
k. Exception.py - Custom exception code
l. Job.py - Python code to initialize jobs based on extracted data
m. Operation.py - Python code initialize operations based on extracted data
n. Pareto_front.py - Contains code to evaluate Pareto solutions from population of solutions
o. Utility.py - Helper functions

5. Petri_net – Contains files to encode designed petri net graph to petri net model
    a. Arc.py - code to encode arcs present in petri net
    b. Machine.py - code to encode machines present in petri net
    c. Model1-pflow – created petri net graph
    d. Petri_net.py - Main file in the directory that transform petri net graph to model
    e. Place.py - code to encode places present in petri net
    f. Stocks.py - code to encode stocks info present in petri net
    g. Transition.py - code to encode transitions present in petri net

6. Rescheduling – Contains implemented rescheduling strategies' files
    a. Complete_rescheduling.py

b. Partial_rescheduling.py
c. Right_shift_rescheduling.py
d. Utility.py - Helper functions

7. Schedule_output – Contains created schedules from experiments
8. Server –
    a. Server.py - code to establish connection with database, does CRUD operations.

        (CRUD) - create, read, update, delete operations on DB

9. algorithms.py - Starting point of algorithm execution while conducting MOO experiments, but not for rescheduling (Simulator.py)
10. Args.py - contains default arguments to conduct rescheduling experiments like algorithm, reschduling method, JSSP and FJSSP problem files path, etc.
11. Change_machine_status.py - File to trigger machine breakdown event while conducting rescheduling experiment.
12. Logger.py - log file to write the experimental logs
13. Net_generation.py - Starting point for petri net model creation
14. Plot_3d.py - used for plotting solution space in 3d.
15. Result_tabulate.py - used to plot rescheduling experimental results
16. Scheduler_date_time_based.py - code to execute production schedule using multi-processing where once process for each job
17. Setup.py - Code and information about software installation (required python packages)
18. Simulation_utility.py - helper function for simulation environment
19. Simulator.py - Main execution point while performing rescheduling experiments