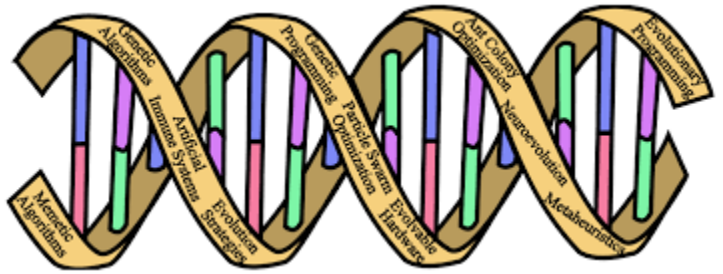
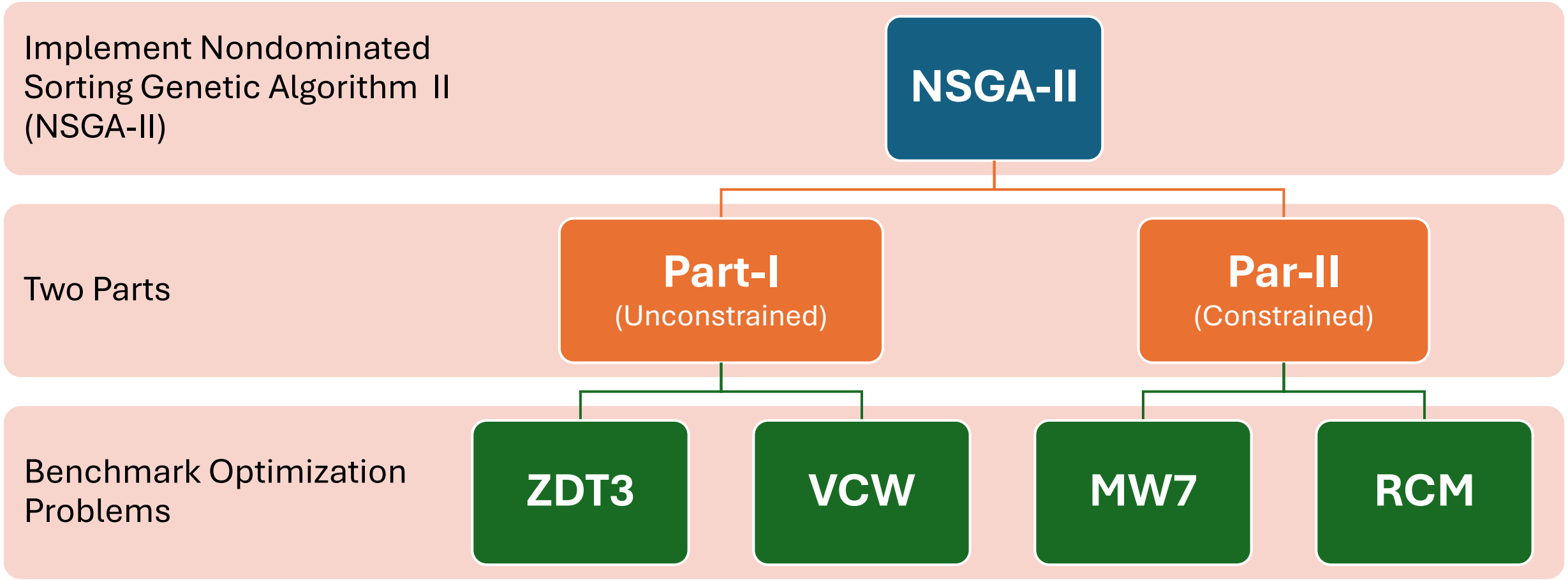


CEG5302

Group Project

Multi-Objective Optimization

Project Structure



Template

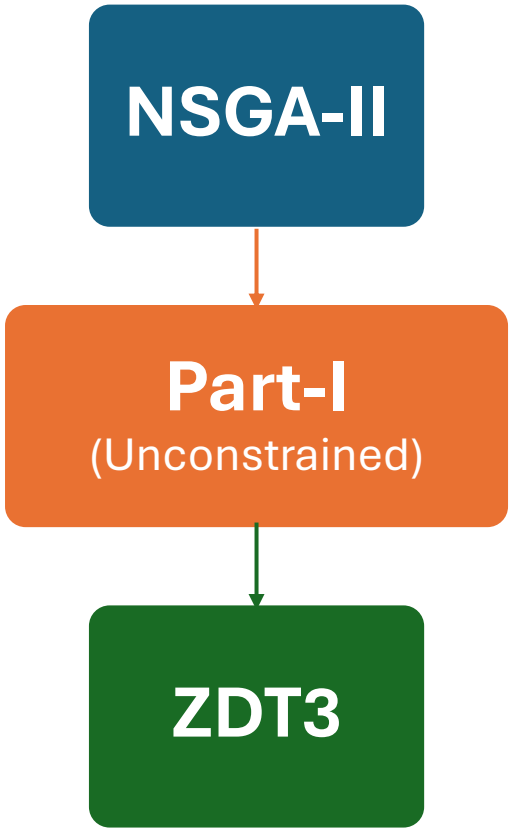
NSGA-II

Part-I

(Unconstrained)

```python
class NSGA2():
    def __init__(self, pop_size=100, ):
        self.pop_size = pop_size
        #-----------BEGIN-----------#
        # TODO: Include other parameters you may use to create an
        #       instance of NSGA2(), such as crossover probability
        #       and mutation probability
        #-----------END-----------#

    def initialize(self, prob):
        x = prob.lower + (prob.upper - prob.lower) * np.random.rand(
            self.pop_size, prob.n_var)
        return x

    def fitness_assignment(self, ):
        #-----------BEGIN-----------#
        # TODO: Implement non-dominated sorting and crowding distance
        # to assign non-domination rank and local crowding distance to each
        # solution in the parent population, which will be used later in
        # tournament selection
        #-----------END-----------#


    def crossover(self, pc):
        #-----------BEGIN-----------#
        # TODO: Implement simulated binary crossover (SBX) to generate
        #       offspring of size `pop_size`
        #-----------END-----------#

    def mutation(self, pm):
        #-----------BEGIN-----------#
        # TODO: Implement polynomial mutation (PM)
        #-----------END-----------
```

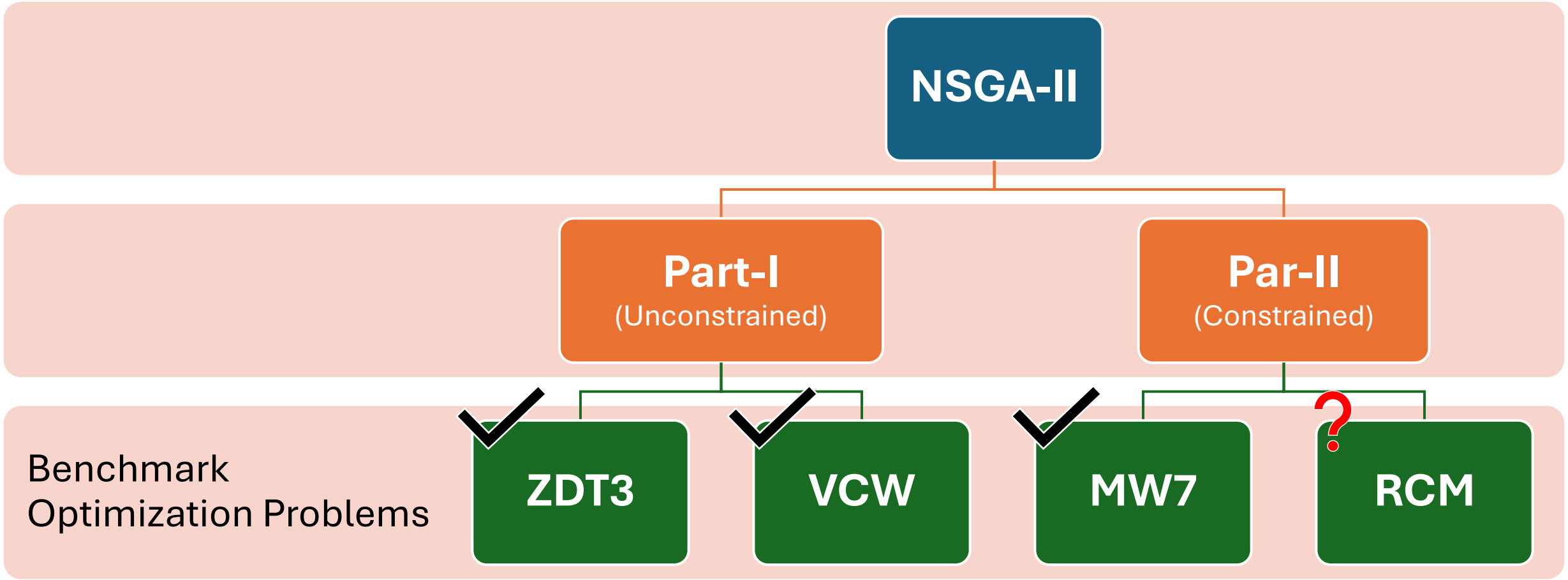
Template




```python
class ZDT3():


    def __init__(self):
        self.name = 'ZDT3'
        self.n_obj = 2  # number of objectives
        self.n_var = 30  # number of decision variables


        self.lower = np.zeros(self.n_var)  # lower bound of decision variables
        self.upper = np.ones(self.n_var)  # upper bound of decision variables


    def evaluate(self, x):
        pop_size = len(x)
        f = np.zeros((pop_size, self.n_obj))
        f[:, 0] = x[:, 0]
        g = 1.0 + 9.0 * np.sum(x[:, 1:], axis=1) / (self.n_var - 1)
        h = 1 - np.power(
            f[:, 0] * 1.0 / g,
            0.5) - (f[:, 0] * 1.0 / g) * np.sin(10 * np.pi * f[:, 0])
        f[:, 1] = g * h


        return f
```

```python
# Instantiate class NSGA2() and class ZDT3()
optimizer = NSGA2(pop_size=100, ??)
problem = ZDT3()


# Use NSGA-II to solve the ZDT3 problem
[optimum_x, optimum_fx] = optimizer.run(problem, max_gen=500)
```

Project Structure



Important Dates

- **Group Formation Deadline:** Mon, Sep 29, 2025 — 11:00 p.m. (SGT) *(Week 7)*
- **Project Release:** Wed, Oct 1, 2025 — 11:00 p.m. (SGT)
- **Project/Report Submission:** Tue, Nov 4, 2025 — 11:00 p.m. (SGT) *(34 days after release)*
- **Project Demo Presentation:** Sat, Nov 8, 2025 (full day) & Mon, Nov 10, 2025 (evening)

Project and Report Submission

Project File:

- The *notebook* must be fully **runnable** **from start to finish**.
- Ensure that all required cells are executed before submission.
- **Do not insert static images**; all results must be generated dynamically.
- **Files with .py extensions will *not be accepted*.**

Project and Report Submission

Report File:

- A report in PDF format with the following specs.
    - Analysis of the algorithms' performance.
    - Insights gained from the experiments; must focus on the reasons for the obtained results.
    - 2-3 page length
    - Single column, Times New Roman, font size 12

Final Deliverables

Submit a single ZIP file named **`EC_Group_<xy>.zip`**, where **`<xy>`** is your group number

(e.g., `EC_Group_22.zip`).

Inside the ZIP include **exactly**:

1. **`report_<xy>.pdf`** *(e.g., `report_22.pdf`)*
2. **`CEG5302_Group_Project_<xy>.ipynb`** *(e.g., `CEG5302_Group_Project_22.ipynb`)*

Use your group number in place of **`<xy>`**. No extra files or folders.

Project Demo

- Tentative demo time distribution per group
    - 15-20 mints
- Mode
    - TBD, could be physical or online

Q & A