Problem Context

Imagine a company that needs to complete a set of 10 tasks within specific deadlines. Each task requires a certain amount of working time, has a predefined difficulty level, a deadline by which it must be completed, and requires a particular skill type. The company has 5 employees, each with a limited number of available hours, a given skill level, and a set of specialized skills. The challenge is to determine which employee should perform which task so that:

1. **Each task is handled by exactly one employee.**
2. **The total work assigned to an employee does not exceed their available working hours.**
3. **An employee's skill level is sufficient for the task's difficulty.**
4. **The employee possesses the specific skill required for the task.**
5. **Deadline considerations are taken into account to avoid potential delays.**

Mathematical Formulation

Decision Variables

Let:

$$T = \{T1, T2, \ldots, T10\} \quad \text{(set of tasks)}$$

$$E = \{E1, E2, \ldots, E5\} \quad \text{(set of employees)}$$

Define the binary decision variable x_{ij} as:

$$x_{ij} = \begin{cases} 1 & \text{if task } i \text{ is assigned to employee } j, \\ 0 & \text{otherwise.} \end{cases}$$

Constraints

1. **Unique Assignment:** Each task must be assigned to one and only one employee:

$$\sum_{j=1}^{5} x_{ij} = 1 \quad \forall\, i \in \{1, \ldots, 10\}.$$

2. **Capacity Constraint:** The total estimated time for tasks assigned to an employee must not exceed their available working hours:

$$\sum_{i=1}^{10} (\text{Time}_i \times x_{ij}) \leq \text{AvailableHours}_j \quad \forall\, j \in \{1, \ldots, 5\}.$$

3. **Skill Level Constraint:** An employee can only be assigned a task if their skill level is at least equal to the task's difficulty:

$$\text{SkillLevel}_j \geq \text{Difficulty}_i \quad \text{whenever } x_{ij} = 1.$$

4. **Specialized Skill Matching:** If a task requires a specific skill (e.g., A, B, or C), it may only be assigned to an employee who possesses that skill.

5. **Deadline Consideration:** An additional penalty is added if an employee's assigned tasks cause the completion time to exceed the task's deadline.

   Hints:

- Sort tasks assigned to an employee by ascending processing time.
- Calculate cumulative finish time f_i after completing the i^{th} task:

$$f_0 = 0, \quad f_i = f_{i-1} + p(i) \quad \text{for } i = 1, 2, \ldots, n.$$

- Check for deadline violations for each task, if its finish time f_i exceeds its deadline $d(i)$, calculate a violation as:

$$\text{Violation}_i = \max\{0, f_i - d(i)\}.$$

For example, an employee is assigned tasks $T1, T3, T5$ and $T9$. Their deadline violation is calculated as follows

Task	Processing Time (hrs)	Deadline (hrs)	Finish Time (hrs)[f_i]	Violation (hrs)	Penalty
T3	2	6	2	$\max\{2-6,0\}=0$	$0\times\gamma$
T9	2	5	4	$\max\{4-5,0\}=0$	$0\times\gamma$
T5	3	7	7	$\max\{7-7,0\}=0$	$0\times\gamma$
T1	4	8	11	$\max\{11-8,0\}=3$	$3\times\gamma$

Table 1: Simple Deadline Violations Calculation

Objective Function

Minimize a cost function defined as:

$$\begin{aligned}\text{Cost} = {} & \alpha \times (\text{Overload Penalty}) + \beta \times (\text{Skill Mismatch Penalty}) \\ & + \delta \times (\text{Difficulty Violation Penalty}) + \gamma \times (\text{Deadline Violation Penalty}) \\ & + \sigma \times (\text{Unique Assignment Violation Penalty})\end{aligned}$$

where $\alpha = 0.20$, $\beta = 0.20$, $\delta = 0.20$, $\sigma = 0.20$ and $\gamma = 0.20$ are weighting factors for the respective penalty terms.

Synthetic Data

Task Data

Task ID	**Estimated Time (hrs)**	**Difficulty**	**Deadline (hrs from now)**	**Required Skill**
T1	4	3	8	A
T2	6	5	12	B
T3	2	2	6	A
T4	5	4	10	C
T5	3	1	7	A
T6	8	6	15	B
T7	4	3	9	C
T8	7	5	14	B
T9	2	2	5	A
T10	6	4	11	C

Table 2: Synthetic Task Data

Employee ID	Available Hours	Skill Level	Skills
E1	10	4	A, C
E2	12	6	A, B, C
E3	8	3	A
E4	15	7	B, C
E5	9	5	A, C

Table 3: Synthetic Employee Data

Employee Data

Approaches for Solving the Problem

- **Genetic Algorithm (GA):**
  - **Encoding:** Represent each solution as a vector where each element indicates the employee assigned to a specific task.
  - **Operators:** Use crossover and mutation to evolve the solution population.
  - **Fitness:** Evaluate solutions based on penalties for any constraint violations.
- **Particle Swarm Optimization (PSO):**
  - **Representation:** Each particle in the swarm represents a candidate assignment.
  - **Update Rule:** Particles update their positions (assignments) based on both individual and group experiences.
  - **Fitness Evaluation:** Similar to GA, using a penalty-based function.
- **Ant Colony Optimization (ACO):**
  - **Representation:** Build solutions incrementally using a graph where nodes represent possible task-employee pairings.
  - **Pheromone Update:** Favor more successful assignments by increasing pheromone levels on beneficial task-employee links.
  - **Solution Construction:** Use probabilistic rules to select assignments.

Submission Contents

Your single ZIP file should contain the following items:

1. **Code Files:**
   - Fully commented and well-organized source code for all three algorithms:
     - Genetic Algorithm (GA)
     - Particle Swarm Optimization (PSO)
     - Ant Colony Optimization (ACO)
   - Each algorithm should be implemented as a standalone module or function. Ensure that the code can be easily executed and tested.
2. **Detailed Report:** The report should include a comprehensive performance evaluation of GA, PSO, and ACO. In your report, provide graphs and discussions for the following performance metrics:
   (a) **Solution Quality (Optimality):**

- **X-axis:** Iterations / Generations / Elapsed Time
- **Y-axis:** Objective (cost) function value (e.g., total penalty)
- *Discussion:* How the solution quality improves as the algorithm runs, including comparisons of the final objective values and convergence behavior.

(b) **Computational Efficiency:**

- **X-axis:** Algorithm type (GA, PSO, ACO) over up to 500 Iterations / Generations
- **Y-axis:** Computational resources used (runtime in seconds, memory usage, etc.)
- *Discussion:* A comparison of how resource consumption scales with problem complexity, and any trade-offs between solution quality and computational cost.

(c) **Constraint Satisfaction (Feasibility):**

- **X-axis:** Iterations / Generations / Elapsed Time
- **Y-axis:** Number (or percentage) of constraint violations (or a feasibility score)
- *Discussion:* Evaluate how quickly each algorithm produces solutions that meet the problem constraints.

Report Structure and Content

Your report should be structured as follows:

Introduction

- **Problem Statement:** Describe the Employee Task Assignment Optimization problem, outlining the key constraints (unique task assignment, capacity limits, skill matching, and deadline adherence) and the objective function (minimizing penalties for overload, skill mismatch, and deadline violations).
- **Purpose:** Explain that the objective of the study is to compare the performance of GA, PSO, and ACO based on multiple performance parameters.

Methodology

- **Algorithm Overview:** Provide a concise description of each algorithm:
  - **Genetic Algorithm (GA):** Explain the encoding, crossover, mutation operators, and how the algorithm is adapted to solve the assignment problem.
  - **Particle Swarm Optimization (PSO):** Describe the representation of solutions, particle updating mechanism, and fitness evaluation.
  - **Ant Colony Optimization (ACO):** Detail the pheromone update process, solution construction mechanism, and how the algorithm selects task-employee assignments.
- **Implementation Details:** Explain:
  - How solutions are encoded (e.g., vector representation for task assignments).
  - How constraints are managed (e.g., using penalty functions or feasibility checks).
  - The parameter settings used in your experiments (e.g., mutation rate, inertia weight, pheromone decay).
- **Experimental Setup:** Include details such as:
  - The synthetic data used (details of tasks and employees).
  - The number of iterations/generations (up to 500) and the number of independent runs to test robustness.
  - The hardware and software environment, if relevant.

Performance Evaluation

For each evaluation category, include the following:

1. **Solution Quality (Optimality)**
   - **Graph:**
     - **X-axis:** Iterations / Generations / Elapsed Time
     - **Y-axis:** Objective (cost) function value (total penalty)
   - **Discussion:** Compare the final objective values and discuss the convergence behavior of each algorithm.
2. **Computational Efficiency**
   - **Graph:**
     - **X-axis:** Algorithm type (GA, PSO, ACO) over up to 500 iterations/generations
     - **Y-axis:** Computational resources used (runtime in seconds, memory usage, etc.)
   - **Discussion:** Analyze runtime performance and memory footprint, and discuss any trade-offs between solution quality and computational cost.
3. **Constraint Satisfaction (Feasibility)**
   - **Graph:**
     - **X-axis:** Iterations / Generations / Elapsed Time
     - **Y-axis:** Number (or percentage) of constraint violations (or a feasibility score)
   - **Discussion:** Evaluate how quickly and effectively each algorithm produces solutions that satisfy the constraints.

Results and Discussion

- **Comparison:** Summarize and compare the performance of GA, PSO, and ACO using the graphs and metrics discussed.
- **Insights:** Analyze the strengths and weaknesses of each algorithm, including the effects of parameter tuning, convergence speed, solution quality, and feasibility.
- **Recommendations:** Based on the analysis, suggest which algorithm(s) may be more suitable for similar task assignment problems and propose potential improvements.

Conclusion

- Summarize the key findings of your performance evaluation.
- Highlight overall performance trends and suggest directions for future work.

Overall Weighting

- **Code:** 40%
- **Report:** 60%

Code (40%)

Criteria	Novice	Competent	Proficient
Functionality	Code is incomplete, fails to run, or produces many errors.	Code runs with minor errors; most functions work as expected but may lack robustness.	Code is fully functional, executes without errors, and meets all assignment requirements.
Code Quality	Code is poorly organized with little to no commenting, making it hard to follow.	Code is organized and commented adequately; structure is logical with minor issues.	Code is exceptionally well-structured and clearly commented; easy to read, understand, and maintain.
Efficiency	Implementation is inefficient with redundant computations and high resource usage.	Code demonstrates moderate efficiency; performance is acceptable but could be optimized further.	Code is highly efficient with optimal resource usage and streamlined logic.

Report (60%)

Criteria	Novice	Competent	Proficient
Content	Report is incomplete, missing key sections and provides only superficial analysis.	Report covers most key sections; analysis is clear but occasionally lacks depth.	Report is comprehensive with all required sections; in-depth analysis and critical insights are provided.
Structure	Report is poorly organized, difficult to follow, and contains numerous grammatical or formatting errors.	Report is reasonably organized with clear sections; minor clarity or formatting issues are present.	Report is excellently structured and clearly written, with a professional presentation and flawless formatting.
Visualizations	Graphs or visuals are missing or poorly labeled, making comprehension difficult.	Visualizations are included and labeled, though they may be inconsistent or lack detail.	Visualizations are high-quality, clearly labeled, and effectively support the analysis.
Discussion	Minimal analysis is provided; discussion lacks critical insights and depth.	Provides a basic analysis with some insights; the discussion is relevant but not fully developed.	Analysis is thorough and critical; discussion is insightful with clear interpretation of results and well-founded recommendations.
Experimental Documentation	Documentation of the experimental setup, parameter settings, and methodology is inadequate or missing.	Documentation is sufficient, covering most aspects of the experimental setup with minor omissions.	Documentation is comprehensive and detailed, clearly describing the experimental setup, parameter settings, and methodology.