Code

```
#JOB SHOP USING ACO\
import numpy as np

class AntColony:
    def __init__(self, num_ants, num_iterations, num_jobs, num_machines, pheromone_init=0.1,
alpha=1, beta=2, evaporation_rate=0.5):
        self.num_ants = num_ants
        self.num_iterations = num_iterations
        self.num_jobs = num_jobs
        self.num_machines = num_machines
        self.pheromone_init = pheromone_init
        self.alpha = alpha
        self.beta = beta
        self.evaporation_rate = evaporation_rate

        # Initialize pheromone matrix
        self.pheromones = np.full((num_jobs, num_jobs), pheromone_init)

    def run(self):
        for iteration in range(self.num_iterations):
            solutions = []
            for ant in range(self.num_ants):
                solution = self.construct_solution()
                solutions.append((solution, self.calculate_cost(solution)))

            # Update pheromones
            self.update_pheromones(solutions)

            # Evaporate pheromones
            self.pheromones *= self.evaporation_rate

            # Choose best solution
            best_solution = min(solutions, key=lambda x: x[1])[0]
            print(f"Iteration {iteration}: Best Solution: {best_solution}")

    def construct_solution(self):
        solution = []
        for job in range(self.num_jobs):
            machine = np.random.randint(0, self.num_machines)
            solution.append((job, machine))
        return solution

    def calculate_cost(self, solution):
```

```
        # Placeholder cost function, to be replaced with actual cost calculation
        return np.random.rand()

    def update_pheromones(self, solutions):
        for solution, cost in solutions:
            for job, machine in solution:
                self.pheromones[job, machine] += 1 / cost

if __name__ == "__main__":
    num_ants = 10
    num_iterations = 20
    num_jobs = 5
    num_machines = 3
    ant_colony = AntColony(num_ants, num_iterations, num_jobs, num_machines)
    ant_colony.run()
```

Output:

```
Iteration 0: Best Solution: [(0, 2), (1, 0), (2, 1), (3, 0), (4, 0)]
Iteration 1: Best Solution: [(0, 2), (1, 1), (2, 1), (3, 1), (4, 0)]
Iteration 2: Best Solution: [(0, 0), (1, 1), (2, 1), (3, 2), (4, 2)]
Iteration 3: Best Solution: [(0, 0), (1, 2), (2, 0), (3, 1), (4, 2)]
Iteration 4: Best Solution: [(0, 0), (1, 1), (2, 1), (3, 0), (4, 2)]
Iteration 5: Best Solution: [(0, 0), (1, 0), (2, 1), (3, 2), (4, 1)]
Iteration 6: Best Solution: [(0, 0), (1, 0), (2, 1), (3, 2), (4, 2)]
Iteration 7: Best Solution: [(0, 2), (1, 1), (2, 1), (3, 0), (4, 2)]
Iteration 8: Best Solution: [(0, 1), (1, 2), (2, 1), (3, 1), (4, 0)]
Iteration 9: Best Solution: [(0, 2), (1, 2), (2, 2), (3, 1), (4, 0)]
Iteration 10: Best Solution: [(0, 2), (1, 1), (2, 0), (3, 0), (4, 1)]
Iteration 11: Best Solution: [(0, 0), (1, 1), (2, 0), (3, 0), (4, 0)]
Iteration 12: Best Solution: [(0, 0), (1, 2), (2, 0), (3, 2), (4, 2)]
Iteration 13: Best Solution: [(0, 2), (1, 0), (2, 0), (3, 2), (4, 2)]
Iteration 14: Best Solution: [(0, 2), (1, 0), (2, 0), (3, 2), (4, 2)]
Iteration 15: Best Solution: [(0, 1), (1, 2), (2, 0), (3, 2), (4, 1)]
Iteration 16: Best Solution: [(0, 2), (1, 0), (2, 1), (3, 1), (4, 2)]
Iteration 17: Best Solution: [(0, 1), (1, 2), (2, 2), (3, 0), (4, 1)]
Iteration 18: Best Solution: [(0, 1), (1, 1), (2, 1), (3, 1), (4, 2)]
Iteration 19: Best Solution: [(0, 0), (1, 2), (2, 2), (3, 2), (4, 0)]
```