CIT645: Algorithmic Problems

Fall 2024

1. Garbage Collection Route Optimization

Description: Minimize fuel usage and emissions by finding efficient routes for waste collection vehicles.
Problem Statement: Given a map of a city with garbage collection points, vehicle capacities, and road distances, compute the optimal routes to minimize total distance traveled.
Sample Input:
Garbage Points: A, B, C, D
Distances: A->B: 2, B->C: 3, A->C: 4, C->D: 2
Vehicle Capacity: 100
Waste Quantities: A: 30, B: 40, C: 20, D: 10

Sample Output:
Route: A -> B -> C -> D -> A
Total Distance: 9

---

2. Water Resource Allocation

Description: Optimize water distribution for agricultural and urban needs in drought-prone areas.
Problem Statement: Allocate water to regions based on demand, minimizing shortages and losses due to pipeline inefficiencies.
Sample Input:
Regions: R1, R2, R3
Water Supply: 1000
Demands: R1: 400, R2: 300, R3: 500
Pipeline Loss: R1: 5%, R2: 3%, R3: 7%

Sample Output:
Allocation: R1: 380, R2: 290, R3: 330

---

3. Smart Grid Task Scheduling

Description: Schedule energy-intensive tasks during renewable energy availability windows.
Problem Statement: Given a day's renewable energy production schedule, allocate tasks to

minimize peak load.
Sample Input:
Tasks: T1: 2kWh, T2: 3kWh, T3: 1kWh
Energy Availability: 9-12: 5kWh, 13-15: 6kWh

Sample Output:
Task Schedule: T1: 9-10, T2: 10-12, T3: 13-14

---

4. Air Pollution Prediction

Description: Predict future air quality based on historical data and traffic trends.
Problem Statement: Develop a model that takes historical air quality data and traffic density as inputs and predicts next-day AQI.
Sample Input:
Historical AQI: 50, 60, 55, 70
Traffic Density: 30, 35, 40, 45

Sample Output:
Predicted AQI: 65

---

5. Carbon Credit Auction System

Description: Optimize the auction process for carbon credits to incentivize emission reductions.
Problem Statement: Given bids from companies, allocate credits to maximize total reduction efficiency.
Sample Input:
Bids: C1: $50 for 10 credits, C2: $45 for 8 credits, C3: $55 for 12 credits
Total Credits: 20

Sample Output:
Allocation: C1: 10 credits, C2: 8 credits, C3: 2 credits
Revenue: $845

---

6. Electric Vehicle (EV) Routing

Description: Optimize routes for EVs considering charging station locations and battery constraints.
Problem Statement: Plan a trip from source to destination, minimizing travel time and ensuring sufficient charge.

Sample Input:
Source: A
Destination: D
Battery Range: 150km
Stations: B: 50km, C: 120km

Sample Output:
Route: A -> C -> D
Total Distance: 170km

---

7. Sustainable Urban Traffic Management

Description: Optimize traffic signal timings to reduce congestion and emissions.
Problem Statement: Adjust signal timings to minimize vehicle wait times at intersections.
Sample Input:
Intersections: I1, I2
Traffic Volume: I1: 300, I2: 500

Sample Output:
Signal Timings: I1: 45s, I2: 60s

---

8. Renewable Energy Mix Optimization

Description: Determine the optimal mix of renewable sources to meet energy demands.
Problem Statement: Allocate solar, wind, and hydro capacities to minimize costs and ensure demand is met.
Sample Input:
Demand: 1000MW
Solar Cost: $50/MW, Wind Cost: $60/MW, Hydro Cost: $40/MW

Sample Output:
Allocation: Solar: 500MW, Wind: 300MW, Hydro: 200MW

---

9. Wildfire Resource Allocation

Description: Allocate firefighting resources to minimize wildfire spread.
Problem Statement: Distribute resources across affected regions based on fire intensity and proximity.
Sample Input:

Regions: R1: High, R2: Medium, R3: Low
Resources: 100 units

Sample Output:
Allocation: R1: 50 units, R2: 30 units, R3: 20 units

---

10. Plastic Waste Recycling Allocation

Description: Allocate recycling plants to process plastic waste efficiently.
Problem Statement: Assign waste collection zones to recycling plants to minimize transport costs.
Sample Input:
Zones: Z1, Z2, Z3
Plants: P1: Capacity 50, P2: Capacity 100
Distances: Z1->P1: 5km, Z1->P2: 8km, Z2->P1: 6km

Sample Output:
Allocation: Z1: P1, Z2: P1, Z3: P2

---

11. University Timetabling Optimization

Description: Optimize the allocation of lectures, halls, and resources to reduce energy consumption and improve accessibility.
Problem Statement: Create a timetable that minimizes conflicts among lectures, maximizes room utilization, and aligns with sustainability goals like reducing lighting and HVAC energy usage in under-occupied rooms.
Sample Input:
Courses: Math101, CS102, Bio201
Professors: P1, P2, P3
Rooms: R1: 50 seats, R2: 100 seats
Time Slots: 9:00-10:00, 10:00-11:00, 11:00-12:00
Constraints:

- Math101 must be scheduled in R1.
- P1 is available only from 10:00-12:00.
- CS102 and Bio201 cannot occur simultaneously.

Sample Output:
Schedule:
9:00-10:00: Math101 in R1 with P1
10:00-11:00: CS102 in R2 with P2

11:00-12:00: Bio201 in R1 with P3
Energy Savings: 20% by optimized room assignments

Deliverables for the Project

1. Code

- A working implementation of the proposed algorithm(s), written in a suitable programming language (e.g., Python, C++, Java).
- The code should be well-documented with comments explaining the logic, functions, and any key sections.

Report Structure (IEEE format - 6-8 pages)

1. Abstract (150-250 words)

Summarize the problem, proposed solution, key findings, and contributions. Mention the evaluation approach and any conclusions.

---

2. Introduction (1 page)

- **Problem Statement**: Briefly describe the societal problem (e.g., university timetabling, water resource allocation) and its importance.
- **Goals**: Outline the algorithm's main objective (e.g., optimize resource allocation).
- **Significance**: Discuss the impact of solving this problem, especially in terms of societal benefits (e.g., improving resource management, reducing pollution).
- **Overview**: Summarize the report’s structure.

---

3. Related Work (1-1.5 pages)

- **Existing Approaches**: Review existing solutions, their methods, strengths, and weaknesses.
- **Comparison**: Explain how these relate to the proposed solution and why it is more effective.

---

4. Proposed Solution (1.5-2 pages)

- **Algorithm Design**: Present the algorithm(s) and justify the approach (e.g., heuristics, optimization techniques).
- **Key Features**: Highlight main features of the solution (e.g., efficiency, conflict minimization).
- **Pseudocode**: Include pseudocode or flowcharts to illustrate the logic.

---

5. Computational Complexity (1 page)

- **Time Complexity**: Analyze how the algorithm scales with problem size.
- **Space Complexity**: Discuss space requirements, if applicable.
- **Comparative Complexity**: Compare with existing solutions, if relevant.

---

6. Proof of Correctness (1 page)

- **Correctness Proof**: Provide formal or logical proof of the algorithm's correctness.
- **Mathematical Justification**: Include any relevant theorems or reasoning to support correctness.

---

7. Implementation (1-1.5 pages)

- **Environment and Tools**: Mention the programming language, libraries, and tools used.
- **Code Structure**: Briefly explain the organization of the code.
- **Challenges**: Discuss challenges faced during implementation and how they were solved.

---

8. Results and Discussion (1.5-2 pages)

- **Experimental Setup**: Describe input data, metrics, and performance criteria used for evaluation.
- **Results**: Present results, including performance comparisons, solution quality, or computational time.
- **Discussion**: Analyze the results, discussing any trade-offs, unexpected findings, and the effectiveness of the solution. Mention limitations and possible improvements.

---

9. Conclusion and Future Work (0.5 page)

- **Summary**: Summarize key findings.

- **Future Work**: Suggest potential improvements or areas for further research.

---

10. References

- List of all cited references formatted in IEEE citation style.

Rubric for Grading the Project (Total Grade: 30)

1. Code (10 points)

- **Correctness (5 points)**:
  Does the code implement the proposed algorithm correctly? Does it produce the expected results for a range of test cases, including edge cases?
- **Clarity and Structure (3 points)**:
  Is the code well-organized, modular, and easy to follow? Are functions and variables named appropriately, reflecting their purpose in a clear and understandable way?
- **Efficiency (2 points)**:
  Does the code handle larger inputs efficiently? Does it perform well within reasonable time and space constraints for the given problem?

---

2. Report (15 points)

Clarity of Report (4 points)

- Is the report structured logically and easy to follow?
- Is the writing concise and clear, avoiding unnecessary complexity or jargon?

Computational Complexity (4 points)

- Is the computational complexity of the proposed algorithm clearly analyzed?
- Does the report include time and space complexity, and is it explained in a straightforward manner?

Proof of Correctness (3 points)

- Does the report provide a concise, logical argument or formal proof to demonstrate that the algorithm is correct?
- Is the correctness argument easy to follow and mathematically sound?

Evaluation of Results (4 points)

- Are the results presented clearly and succinctly?
- Does the report provide a focused analysis of the results, including performance comparison, trade-offs, and limitations?