Set Covering Problem

PIA: Part II

Operations Research, 032
Team 6: Alan Alejandro Vargas González, 2086183

May, 2025

1 Part II: Codification and Visualization

1.1 Model Codification

The model was implemented in Python using the PuLP library. The complete code is stored in the set-covering-repository.

```
def solving(m, n, costs, cov):
    prob = pulp.LpProblem("Set_Covering_Problem", pulp.LpMinimize)

    x = [pulp.LpVariable(f"x_{j}", cat="Binary") for j in range(n)]

    # Objective function
    prob += pulp.lpSum([costs[j] * x[j] for j in range(n)])

    # Constraints
    for i in range(m):
        prob += pulp.lpSum([x[j] for j in cov[i]]) >= 1

    start = time.time()
    prob.solve()
    amount_time = time.time() - start

    subset = [j for j in range(n) if pulp.value(x[j]) == 1]
    total_cost = pulp.value(prob.objective)

    return subset, total_cost, amount_time
```

1.2 Data Structures

The next data structures were used:

- List `costs`: contains the cost of each subset.
- List `coverage` lists: contains, for each element, the subsets that cover it.
- Binary variable `x[j]`: is the j subset selected?

1.3 Data Visualization

Validation of:

- All the costs are positive.
- Each row is cover by at least one subset.
- There aren't indexes out of range.

```
def validation(m, n, costs, cov):
    assert len(costs) == n, "Length of costs don't match n"
    assert len(cov) == m, "Length of coverage don't match m"
    for c in costs:
        assert c > 0, "Costs need to be positive"
    for i in range(m):
        assert len(cov[i]) > 0, f"Row {i} doesn't have coverage"
        for col in cov[i]:
            assert 0 <= col < n, f"Index out of range in coverage of row {i}"
```

1.4 Preliminary Analysis

Testing with small data, such as the `scp's` instances, obtaining consistent results with known values. I created a small example to verify the correct functionality of the program:

- **Elements:** 3
- **Subsets:** 4
- **Costs:** $[1, 2, 3, 4]$
- **Coverage:**
  - Row 0: covered by subsets $\{0, 1\}$
  - Row 1: covered by subsets $\{1, 2\}$
  - Row 2: covered by subsets $\{2, 3\}$

Results:

- **Subsets selected:** $[0, 2]$
- **Total cost:** 4
- **Execution time:** 0.0784 s

Why these subsets?

- **Row 0** is covered by subsets $[0, 1]$ → Covered by subset 0.
- **Row 1** is covered by subsets $[1, 2]$ → Covered by subset 2.
- **Row 2** is covered by subsets $[2, 3]$ → Covered by subset 2.

Therefore, the subsets 0 and 2 are enough to cover all rows.

Total cost The total cost is:

$$\text{Total Cost} = \text{cost}[0] + \text{cost}[2] = 1 + 3 = 4.$$

Execution time The solver took **0.1287s** to find the optimal solution.