Lexicographic Min-Max Fairness in Task Assignment

Abhinav V S (22008) Chaitanya Shinde (22085) Dev Raj Singh (22102)

April 24, 2025

1. Introduction

Traditional assignment problems focus on minimizing total cost or maximizing utility. In multi-agent systems, fairness becomes critical to avoid overburdening specific agents. Lexicographic Min-Max Fairness (Lexifairness) goes beyond min-max fairness by minimizing the maximum cost, then the second-highest, and so on.

Key Contributions:

- Algorithms for one-to-one and one-to-many assignments.
- Tractable approximations for computationally hard problems.
- Efficiency-fairness trade-off analysis using experiments.

2. Objective Function

The optimization problem is modeled as:

$$\begin{aligned}
\min \quad & z = \sum_i \sum_j c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_i x_{ij} = 1, \quad \forall j \\
& \sum_j x_{ij} = 1, \quad \forall i \\
& c_{ij} x_{ij} \leq z, \quad \forall i, j \\
& x_{ij} \in \{0, 1\}
\end{aligned}$$

Lexifairness modifies this by iteratively minimizing the cost vector in descending order.

3. Constraints

- Each task assigned to one agent: $\sum_i x_{ij} = 1$
- Each agent does one task: $\sum_j x_{ij} = 1$
- Task costs respect fairness: $c_{ij} x_{ij} \leq z$

Price of Fairness (PoF):

$$\mathrm{PoF}(c) = \frac{\sum_{i,j} c_{ij} x_{ij}^{\text{fair}} - \sum_{i,j} c_{ij} x_{ij}^{\text{eff}}}{\sum_{i,j} c_{ij} x_{ij}^{\text{eff}}}$$

4. Algorithms

4.1 Iterative Min-Max Fair Assignment

1. Solve MILP for min-max fairness.
2. Fix the agent-task pair with the maximum cost.
3. Update and repeat.

Pros: Simple, uses PuLP in Python. **Cons:** Expensive for large n due to repeated MILPs.

4.2 Network Flow-Based Lexifair Assignment

1. Build bipartite graph of agents and tasks.
2. Add edges in increasing cost order.
3. Use max-flow algorithm to solve feasibility.
4. Fix bottleneck edges iteratively.

Pros: Polynomial time, scalable, uses NetworkX.

5. Implementation Details

MILP Approach: - PuLP, Python. - Solves MILP per iteration.
Network Flow: - NetworkX, Python. - Graph-based max-flow/min-cut.

6. Results Example

Input Cost Matrix:

$$\begin{bmatrix} 9 & 5 & 6 \\ 8 & 2 & 4 \\ 7 & 3 & 1 \end{bmatrix}$$

Assignments:

- Efficient: [9, 2, 1]
- Min-Max Fair: [7, 2, 6]
- Lexifair: [7, 5, 4]

7. Evaluation and Gini Coefficient

Gini Coefficient: Measures inequality. - $G = 0$: Perfect fairness - $G = 1$: Complete inequality
Lexifairness assignments show reduced Gini and better cost distribution compared to efficient or min-max only assignments.

8. Practical Applications

- Job assignment with differing worker capabilities.
- Emergency shelter allocation minimizing distance disparity.
- Load balancing in cloud computing.
- Fair assignment in ridesharing, crowdsourcing.

9. Conclusion

Lexifairness ensures balanced agent workloads at some cost to efficiency. The combination of MILP and network flow methods allows for practical, scalable implementations. The approach is well-suited for multi-agent, human-involved, and high-stakes environments.

10. References

- Geoffrey Ding and Hamsa Balakrishnan, "Lexicographic Min-Max Fairness in Task Assignments", CDC 2023.