IIT ROPAR

CAPSTONE 1

MID TERM PRESENTATION

Ananya Sethi 2022CSB1066
Utkarsh Patel 2022CSB1140

Dr. Shweta Jain

INTRODUCTION

We began our BTP by reviewing several research papers on Multi-Armed Bandits and many-to-many matching problems.

Multi-Armed Bandits

1 **Uncertainty in Rewards**: Each option (or "arm") provides an unknown reward, requiring careful sampling to understand potential outcomes.

2 **Exploration vs. Exploitation:** The agent must balance trying new arms to discover their rewards (exploration) with choosing the best-known option to maximize immediate gains (exploitation).

3 **Sequential Learning:** Decisions are made over time, allowing the agent to update its strategy based on past outcomes to improve long-term performance.

Many-to-Many matching

4 **Dual-Sided Assignments:** Involves pairing agents from two groups, where each agent can be matched with multiple partners, reflecting a two-sided market.

5 **Capacity and Constraints:** Matches are subject to limitations, such as the maximum number of partnerships an agent can handle or specific resource constraints.

6 **Preference and Fairness:** Both groups often have preferences over potential matches, and the goal is to achieve fair, efficient assignments that best satisfy these mutual preferences.

Key Problems

By observing the above two problems, we developed the idea of how to make allocations in situations where supervisors have limited resources—essentially a many-to-one matching problem—and maintain their preferences for students. In our approach, when multiple students are allocated to the same supervisor, the more preferred students receive a larger share of that supervisor's resources.

UAV Bandwidth Allocation with Collisions:

Agents aim to secure maximum bandwidth from UAVs. However, when multiple agents choose the same UAV, the bandwidth is equally divided among them, creating challenges due to resource sharing and potential collisions.

Supervisor-Student Matching:

This problem matches supervisors and students, each with their own preferences, while supervisors have limited slots. A modified Gale-Shapley algorithm is used to achieve stable matchings that honor both preferences and resource constraints.

Each supervisor s_i has a preference ordering over the agents and is denoted by:

$$a^i_{p_1} \succ a^i_{p_2} \succ \cdots \succ a^i_{p_n},$$

where $a^i_{p_1}$ is the most preferred and $a^i_{p_n}$ is the least preferred agent for supervisor i. The weight of supervisor s_i for an agent a_j is defined as:

$$w_{i,j} = \gamma^{\text{rank}_i(a_j)}, \quad \gamma < 1,$$

where $\text{rank}_i(a_j)$ denotes the position of a_j in the preference list of s_i.

Let S_i denote the set of agents assigned to supervisor s_i:

$$S_i = \{a_j \in A \mid x_{i,j} = 1\},$$

$x_{i,j}$ is a binary decision variable:

$$x_{i,j} = \begin{cases} 1, & \text{if agent } a_j \text{ is assigned supervisor } s_i, \\ 0, & \text{otherwise.} \end{cases}$$

The utility received by agent a_j when assigned supervisor s_i is:

$$U_{i,j} = \frac{w_{i,j} v_i}{\sum\limits_{a_m \in S_i} w_{i,m}}.$$

each agent receives exactly one supervisor, the overall utility of agent a_j is:

$$U_j(x) = \sum_{i=1}^{k} x_{i,j} U_{i,j}.$$

For any two agents a_j and a_m, the envy of agent a_j toward agent a_m is defined as:

$$E_{j,m}(x) = \max\{0, U_m(x) - U_j(x)\}.$$

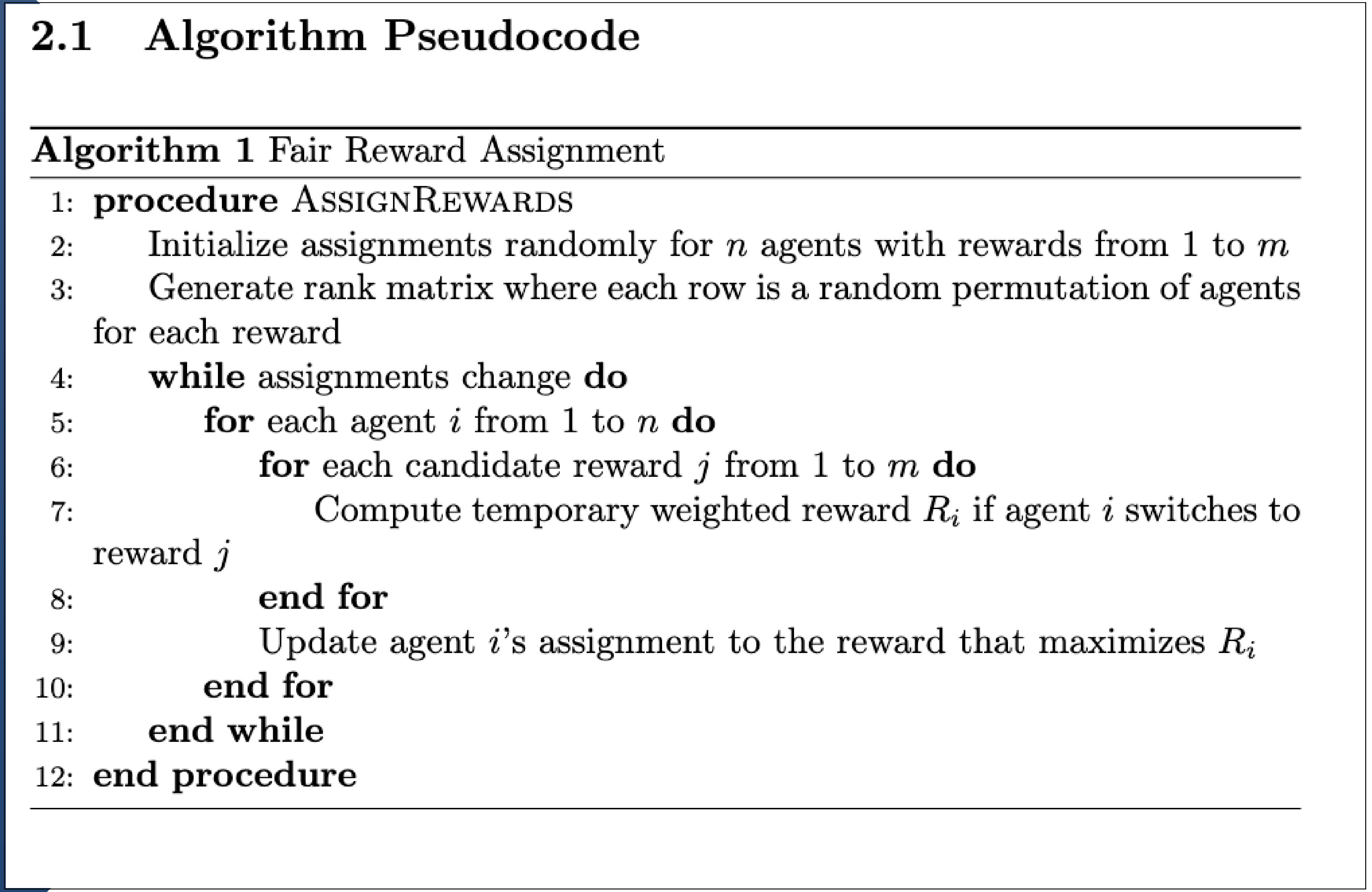
OFFLINE PS

The primary objective of our research is to find an assignment of supervisors to agents that yields a stable solution (Nash Equilibrium) such that it minimizes the maximum envy between any two agents

$$\boxed{\min_x \max_{j,m} E_{j,m}(x),}$$

INITIAL ALOGRITHM

-In this initial algorithm, each agent attempts to incrementally increase its envy by one unit until no further improvement is possible.



2.1 Algorithm Pseudocode

Algorithm 1 Fair Reward Assignment

```
procedure AssignRewards
    Initialize assignments randomly for $n$ agents with rewards from 1 to $m$
    Generate rank matrix where each row is a random permutation of agents for each reward
    while assignments change do
        for each agent $i$ from 1 to $n$ do
            for each candidate reward $j$ from 1 to $m$ do
                Compute temporary weighted reward $R_i$ if agent $i$ switches to reward $j$
            end for
            Update agent $i$'s assignment to the reward that maximizes $R_i$
        end for
    end while
end procedure
```

Problem here:

The problem encountered during the implementation was that, although multiple simulations indicated convergence and decreasing envy, the convergence was not guaranteed. Moreover, despite our goal of minimizing envy, the agents focused on maximizing their rewards.

Congestion Games

A type of game where agents choose from a set of resources, and the cost (or utility) associated with a resource depends on how many agents are using it. Each agent's decision affects the overall performance of the system, and the existence of a potential function guarantees at least one pure strategy Nash equilibrium.

Potential Function

A scalar function Φ defined over the strategy profiles of all agents such that any unilateral change by an agent results in a corresponding change in Φ that mirrors the change in that agent's utility. This property helps ensure that iterative improvements lead to convergence towards equilibrium.

Observation 1

For finite potential games all executions of the best response dynamics terminate

Observation 2

Every congestion game is a potential game.

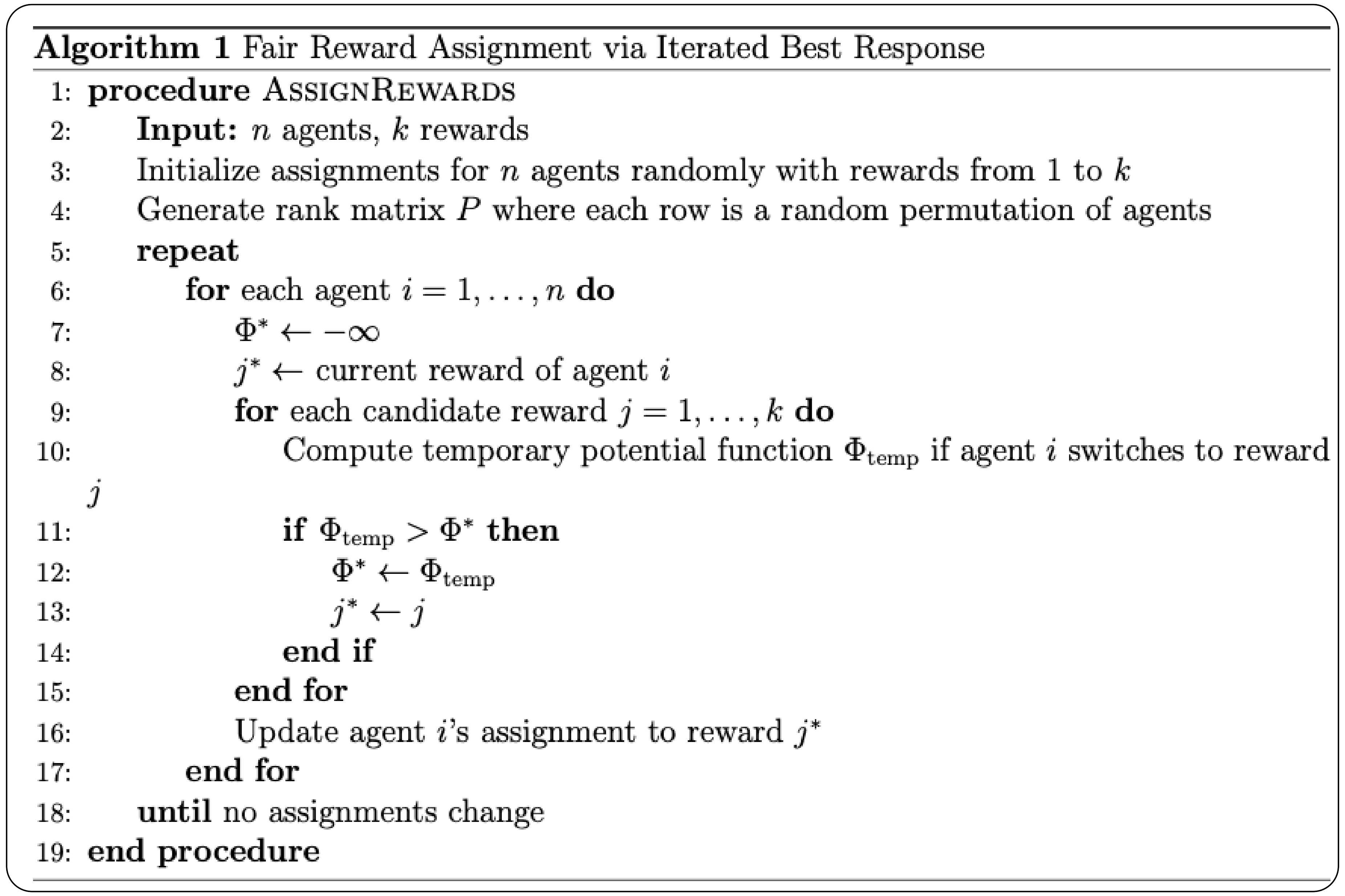
Observation 3

Every CG has a Nash equilibrium in pure strategies. This can be shown by constructing a potential function that assigns a value to each outcome. Moreover it is shown that iterated best response finds a Nash equilibrium

BTP
Evaluations

ALGORITHM

Mission Statement



Algorithm 1 Fair Reward Assignment via Iterated Best Response

```
procedure AssignRewards
    Input: $n$ agents, $k$ rewards
    Initialize assignments for $n$ agents randomly with rewards from 1 to $k$
    Generate rank matrix $P$ where each row is a random permutation of agents
    repeat
        for each agent $i = 1, \dots, n$ do
            $\Phi^* \leftarrow -\infty$
            $j^* \leftarrow$ current reward of agent $i$
            for each candidate reward $j = 1, \dots, k$ do
                Compute temporary potential function $\Phi_{\text{temp}}$ if agent $i$ switches to reward $j$
                if $\Phi_{\text{temp}} > \Phi^*$ then
                    $\Phi^* \leftarrow \Phi_{\text{temp}}$
                    $j^* \leftarrow j$
                end if
            end for
            Update agent $i$'s assignment to reward $j^*$
        end for
    until no assignments change
end procedure
```

Vision Statement

This algorithm fairly assigns k rewards to n agents using an **iterative best response strategy** to minimize envy.

Steps:

1. Initialize random reward assignments and a rank matrix (P) with agent preferences.
    - Iterate Until Stable: For each agent, evaluate all possible reward switches.
    - Compute a potential function (Φ) to measure fairness.
    - Update the assignment if a better reward is foun
2. Stop when no agent wants to switch, ensuring a fair and stable allocation.

Key Idea:

Our approach is inspired by **congestion games** and **potential functions**, ensuring fairness and efficiency in resource allocation as supervisors have rewards associated with them and students want to maximize their utility by getting the best rewards.

$$\boxed{\Phi(r) = \sum_{i=1}^{n} \ln(U_i)}$$

Maximization of $\sum \ln(U_j)$ Ensures Minimization of Envy

Related Proofs

Theorem 1. *Maximizing the function*

$$F = \sum_{j=1}^{n} \ln(U_j)$$

ensures that the envy

$$E_{j,m} = \max\{0, U_m - U_j\}.$$

is minimized.

Proof. We show that maximizing F leads to an equal allocation of utilities, which in turn minimizes envy. By the Arithmetic Mean-Geometric Mean (AM-GM) inequality, for positive numbers $U_1, U_2, \ldots, U_n$ we have

$$\prod_{j=1}^{n} U_j \leq \left(\frac{1}{n} \sum_{j=1}^{n} U_j \right)^n,$$

with equality if and only if $U_1 = U_2 = \cdots = U_n$. Taking the natural logarithm of both sides yields

$$\sum_{j=1}^{n} \ln(U_j) \leq n \ln \left(\frac{1}{n} \sum_{j=1}^{n} U_j \right),$$

with equality if and only if all U_j are equal. Thus, under any fixed total (or average) utility constraint, F is maximized when

$$U_1 = U_2 = \cdots = U_n = c,$$

for some constant c. In this scenario, for any pair (j, m) we have

$$\max(0, U_m - U_j) = \max(0, c - c) = 0.$$

Therefore, the envy is

$$E_{\mathrm{j,m}} = 0,$$

which is the minimum possible value. Hence, maximizing $\sum_{j=1}^{n} \ln(U_j)$ leads to the minimization of envy.

5.2 Proof of Algorithm Termination

Related Proofs

Proof. We prove that the fair reward assignment algorithm terminates by showing that the potential function

$$\Phi(r) = \sum_{i=1}^{n} \ln(U_i)$$

(where U_i is the utility of agent a_i) strictly increases with each agent's best response update and that there exists an upper bound on this function.

Let S_j^t and S_k^t be the sets of agents assigned to rewards r_j and r_k, respectively, at iteration t. After agent a_i moves, these sets update to:

$$S_j^{t+1} = S_j^t \setminus \{a_i\}, \quad S_k^{t+1} = S_k^t \cup \{a_i\}.$$

The change in the potential function is given by

$$\begin{aligned}\Delta\Phi &= \Phi(r^{t+1}) - \Phi(r^t) \\ &= \left[\sum_{a \in S_j^{t+1}} \ln(U_a) - \sum_{a \in S_j^t} \ln(U_a)\right] + \left[\sum_{a \in S_k^{t+1}} \ln(U_a) - \sum_{a \in S_k^t} \ln(U_a)\right]. \qquad (8)\end{aligned}$$

Since agent a_i performs a best response update, the increase in its utility guarantees that $\ln(U_i)$ increases; hence, $\Delta\Phi > 0$ unless a_i already has a best response (in which case $\Delta\Phi = 0$). The algorithm terminates when no agent can further improve its utility, i.e., when $\Delta\Phi = 0$ for all agents, corresponding to a stable allocation.

Future Work

1. Prove the theorems in support of our algorithm like proof of
   - Ensuring no supervisor remains unassigned, etc.
   - Bound on envy
2. Decentralized Multi-Player Multi-Armed Bandit Framework
   - Extend the model where rewards and preference orders are unknown.
   - Agents learn optimal allocations over time.
3. Stochastic Rewards & Collisions
   - Arms provide random rewards, requiring exploration.
   - Collisions lead to proportional reward sharing instead of strict competition.
4. Agent Communication & Coordination
   - Enable communication to reduce collisions and improve decision-making.
   - Drive the system toward a stable matching that respects agent and arm preferences.

DR. SHWETA JAIN

End Sem Presentation

Ananya Sethi
Utkarsh Patel

Complexity Analysis

Algorithm 1 has a worst-case time complexity of O(n^2kT), where n is the number of agents, k is the number of rewards, and T is the number of iterations until convergence and space complexity of O(nk).

In each iteration, the algorithm performs the following operations:
• For each of the n agents, the algorithm evaluates k possible reward reassignments.
• For each evaluation, the algorithm needs to compute the new reward distribution, which takes O(n) time to account for all agents sharing the same reward. Therefore, each iteration has a time complexity of O(n^2k). Let T be the number of iterations until convergence. The worst-case time complexity of the algorithm is then O(n^2kT).

For space complexity, the algorithm requires:
• O(nk) space for storing the weight matrix wi,j
• O(n) space for current reward assignments
• O(k) space for tracking agents assigned to each reward
Therefore, the overall space complexity is O(nk).

Relationship between Nash Social Welfare and Envy

As the NSW increases, the maximum pairwise envy between agents monotonically decreases.

NSW Defination

$$\mathrm{NSW}(x) = \prod_{j=1}^{n} U_j(x)$$

AM-GM Inequality

$$\frac{1}{n}\sum_{j=1}^{n} U_j \geq \left(\prod_{j=1}^{n} U_j\right)^{1/n}$$

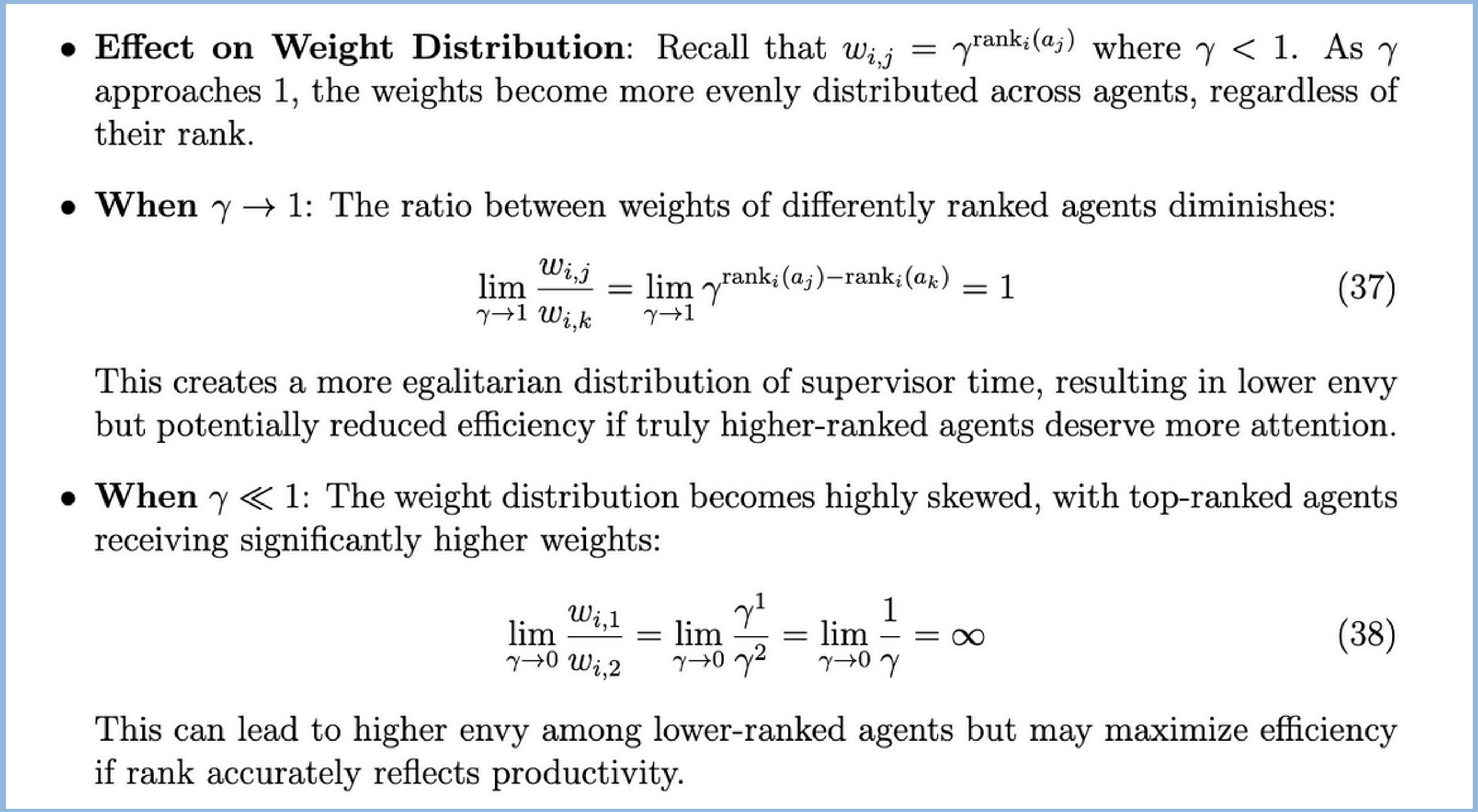
Logarthmic Transformation of NSW

$$\log(\mathrm{NSW}) = \sum_{j=1}^{n} \log(U_j)$$

Conclusion

$$\mathrm{NSW}_2 > \mathrm{NSW}_1 \implies \max(E_{j,m} \text{ in } \mathrm{NSW}_2) \leq \max(E_{j,m} \text{ in } \mathrm{NSW}_1)$$

Role of Gamma in Nash Social Welfare and Envy



- **Effect on Weight Distribution**: Recall that $w_{i,j} = \gamma^{\text{rank}_i(a_j)}$ where $\gamma < 1$. As γ approaches 1, the weights become more evenly distributed across agents, regardless of their rank.
- **When** $\gamma \to 1$: The ratio between weights of differently ranked agents diminishes:

$$\lim_{\gamma \to 1} \frac{w_{i,j}}{w_{i,k}} = \lim_{\gamma \to 1} \gamma^{\text{rank}_i(a_j) - \text{rank}_i(a_k)} = 1 \tag{37}$$

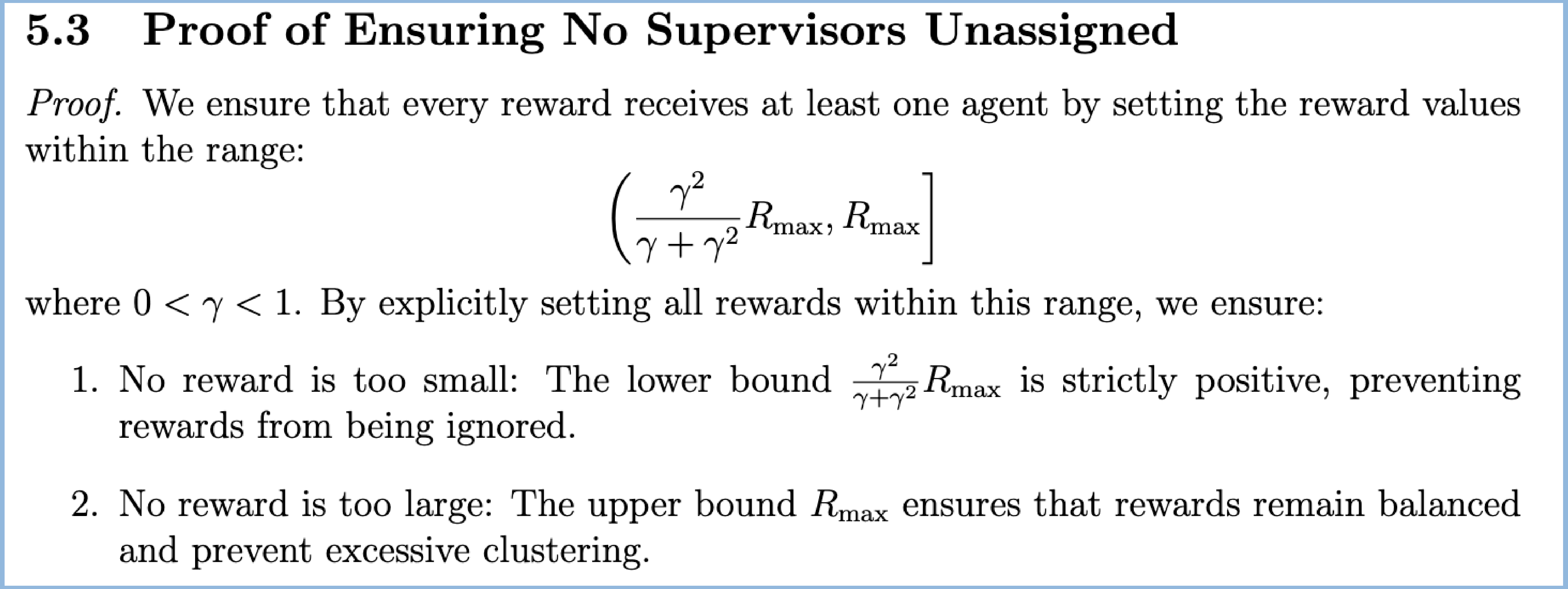
  This creates a more egalitarian distribution of supervisor time, resulting in lower envy but potentially reduced efficiency if truly higher-ranked agents deserve more attention.

- **When** $\gamma \ll 1$: The weight distribution becomes highly skewed, with top-ranked agents receiving significantly higher weights:

$$\lim_{\gamma \to 0} \frac{w_{i,1}}{w_{i,2}} = \lim_{\gamma \to 0} \frac{\gamma^1}{\gamma^2} = \lim_{\gamma \to 0} \frac{1}{\gamma} = \infty \tag{38}$$

  This can lead to higher envy among lower-ranked agents but may maximize efficiency if rank accurately reflects productivity.

In student-supervisor matching specifically, γ can be interpreted as determining how supervisor time is distributed among students of different ranks. When γ is closer to 1, supervisor time is more evenly distributed, reducing envy but potentially not optimizing productivity. When γ is smaller, supervisor time is concentrated among higher-ranked students, potentially optimizing productivity but at the cost of higher envy among lower-ranked students.



5.3 Proof of Ensuring No Supervisors Unassigned

Proof. We ensure that every reward receives at least one agent by setting the reward values within the range:

$$\left(\frac{\gamma^2}{\gamma + \gamma^2} R_{\max}, R_{\max}\right]$$

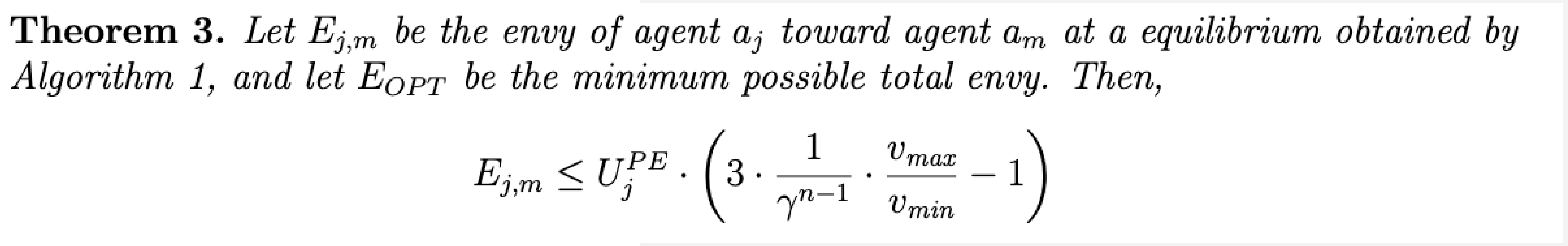
where $0 < \gamma < 1$. By explicitly setting all rewards within this range, we ensure:

1. No reward is too small: The lower bound $\frac{\gamma^2}{\gamma+\gamma^2} R_{\max}$ is strictly positive, preventing rewards from being ignored.

2. No reward is too large: The upper bound $R_{\max}$ ensures that rewards remain balanced and prevent excessive clustering.

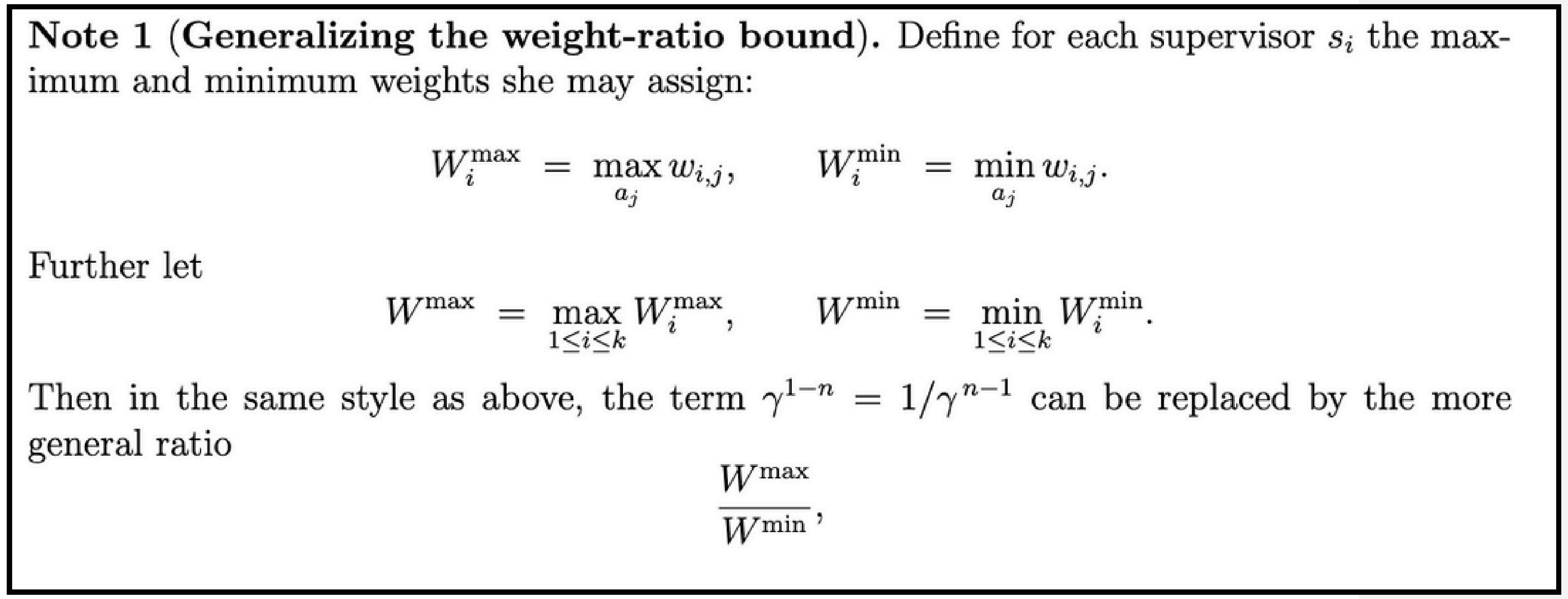
- Constant Sum Game
- Pareto-Optimality

BOUND ON ENVY



Theorem 3. *Let $E_{j,m}$ be the envy of agent a_j toward agent a_m at a equilibrium obtained by Algorithm 1, and let E_{OPT} be the minimum possible total envy. Then,*

$$E_{j,m} \leq U_j^{PE} \cdot \left(3 \cdot \frac{1}{\gamma^{n-1}} \cdot \frac{v_{max}}{v_{min}} - 1\right)$$



Note 1 (Generalizing the weight-ratio bound). Define for each supervisor s_i the maximum and minimum weights she may assign:

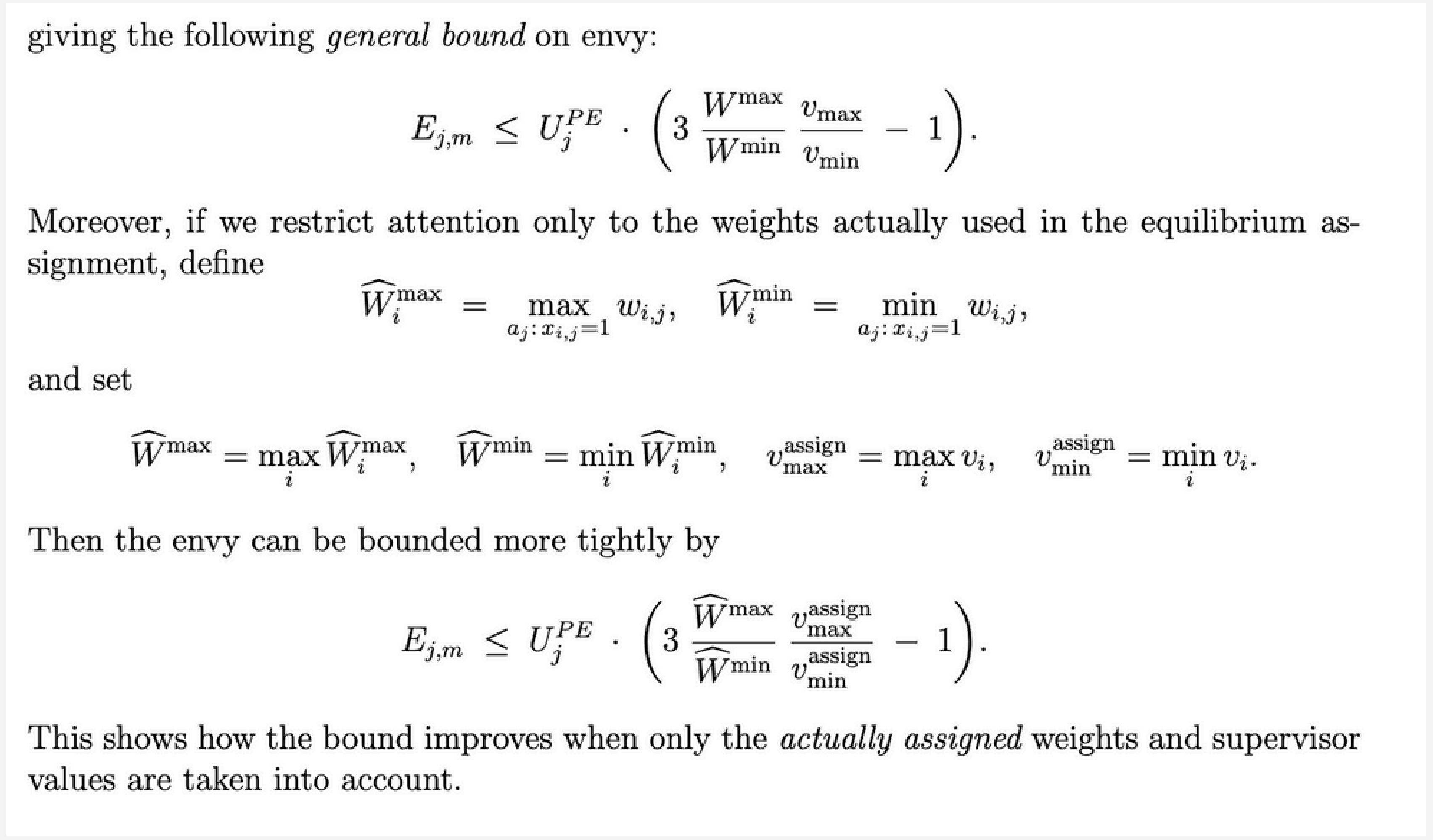
$$W_i^{\max} = \max_{a_j} w_{i,j}, \qquad W_i^{\min} = \min_{a_j} w_{i,j}.$$

Further let

$$W^{\max} = \max_{1 \leq i \leq k} W_i^{\max}, \qquad W^{\min} = \min_{1 \leq i \leq k} W_i^{\min}.$$

Then in the same style as above, the term $\gamma^{1-n} = 1/\gamma^{n-1}$ can be replaced by the more general ratio

$$\frac{W^{\max}}{W^{\min}},$$



giving the following *general bound* on envy:

$$E_{j,m} \leq U_j^{PE} \cdot \left(3 \frac{W^{\max}}{W^{\min}} \frac{v_{\max}}{v_{\min}} - 1\right).$$

Moreover, if we restrict attention only to the weights actually used in the equilibrium assignment, define

$$\widehat{W}_i^{\max} = \max_{a_j : x_{i,j}=1} w_{i,j}, \quad \widehat{W}_i^{\min} = \min_{a_j : x_{i,j}=1} w_{i,j},$$

and set

$$\widehat{W}^{\max} = \max_i \widehat{W}_i^{\max}, \quad \widehat{W}^{\min} = \min_i \widehat{W}_i^{\min}, \quad v_{\max}^{\text{assign}} = \max_i v_i, \quad v_{\min}^{\text{assign}} = \min_i v_i.$$

Then the envy can be bounded more tightly by

$$E_{j,m} \leq U_j^{PE} \cdot \left(3 \frac{\widehat{W}^{\max}}{\widehat{W}^{\min}} \frac{v_{\max}^{\text{assign}}}{v_{\min}^{\text{assign}}} - 1\right).$$

This shows how the bound improves when only the *actually assigned* weights and supervisor values are taken into account.

End Sem Presentation

5.6 Analysis of Maximum Utility Gain at Equilibrium

Related Proofs

$$U_{max} = \frac{w_{k,j} \cdot v_k}{\sum_{a_l \in S_k} w_{k,l} + w_{k,j}} - \frac{w_{k,m} \cdot v_k}{\sum_{a_l \in S_k} w_{k,l} \cdot 3 \cdot \frac{1}{\gamma^{n-1}} \cdot \frac{v_{max}}{v_{min}}}$$

For simplicity, let's define:

$$W_k = \sum_{a_l \in S_k} w_{k,l} \tag{29}$$

$$\alpha = 3 \cdot \frac{1}{\gamma^{n-1}} \cdot \frac{v_{max}}{v_{min}} \tag{30}$$

Then:

$$U_{max} = v_k \left[\frac{w_{k,j}}{W_k + w_{k,j}} - \frac{w_{k,m}}{W_k \cdot \alpha} \right]$$

This formula reveals several insights:

1. The potential gain scales linearly with v_k, indicating that higher-value supervisors offer greater potential improvement.

2. The gain depends on the ratio of agent a_j's weight $(w_{k,j})$ versus agent a_m's weight $(w_{k,m})$ for supervisor s_k.

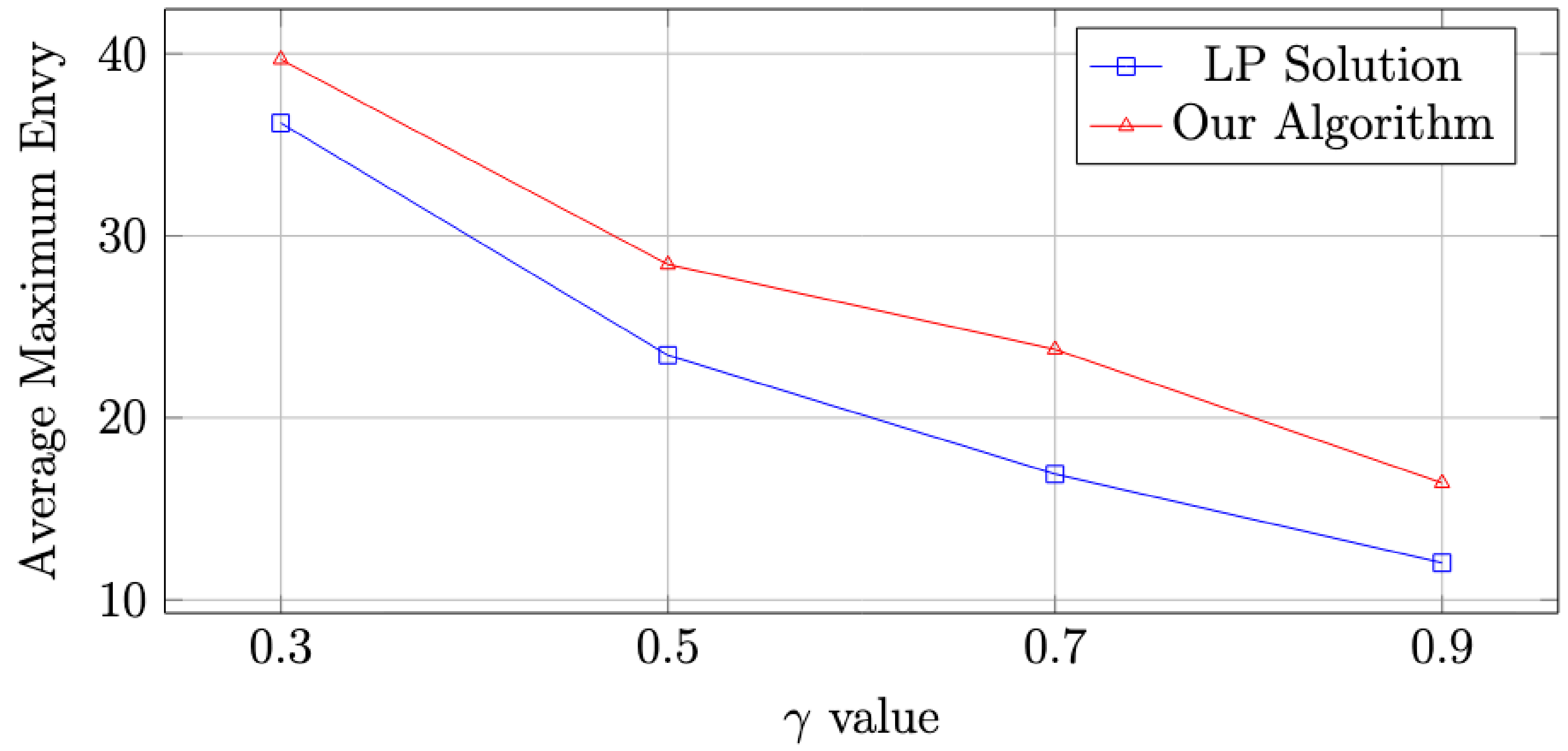
3. The denominator $W_k + w_{k,j}$ represents the "congestion" effect after shifting. As W_k increases, the potential gain decreases.

4. The bound coefficient α increases with larger n, smaller γ, and larger ratio $\frac{v_{max}}{v_{min}}$.

5. At a true equilibrium, U_{max} should be non-positive, confirming that no agent can unilaterally improve their utility.

BTP
Evaluation

Results



01 https://github.com/utkarshpatel7356/BTP_Paper1

- **Effect of γ:** As γ increases (weaker preference decay), maximum envy decreases substantially for both methods. With $\gamma = 0.3$, the average maximum envy is about three times higher than with $\gamma = 0.9$, indicating that stronger preference distinctions lead to significantly higher envy levels.

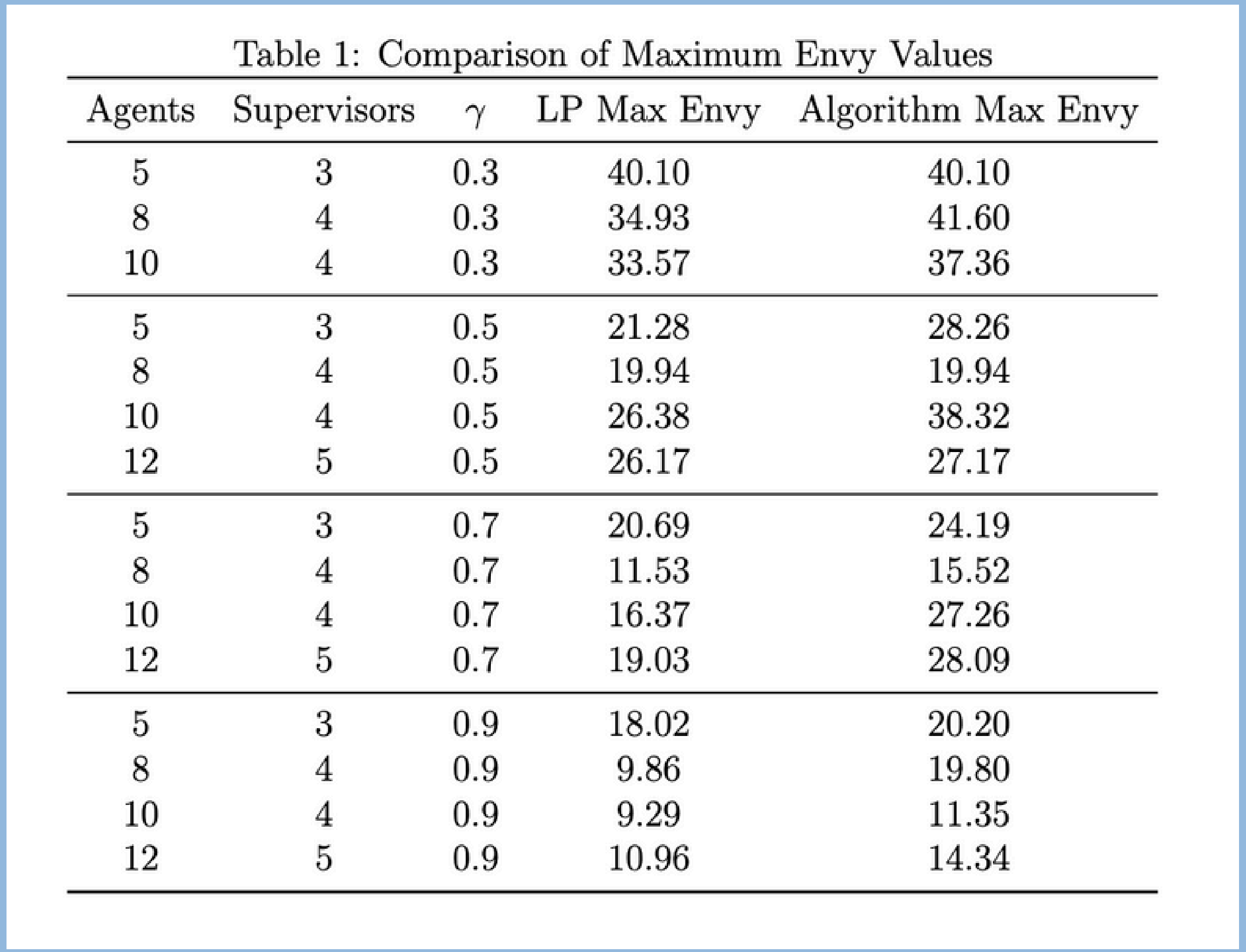


Table 1: Comparison of Maximum Envy Values

Agents	Supervisors	γ	LP Max Envy	Algorithm Max Envy
5	3	0.3	40.10	40.10
8	4	0.3	34.93	41.60
10	4	0.3	33.57	37.36
5	3	0.5	21.28	28.26
8	4	0.5	19.94	19.94
10	4	0.5	26.38	38.32
12	5	0.5	26.17	27.17
5	3	0.7	20.69	24.19
8	4	0.7	11.53	15.52
10	4	0.7	16.37	27.26
12	5	0.7	19.03	28.09
5	3	0.9	18.02	20.20
8	4	0.9	9.86	19.80
10	4	0.9	9.29	11.35
12	5	0.9	10.96	14.34

Supervisor Utility

To capture the preferences and load-balancing considerations of supervisors in our allocation mechanism, we introduce a *supervisor utility* term. This additional component aligns the objectives of agents and supervisors within a unified potential framework, ensuring that supervisors prefer assignments with fewer, better-ranked agents.

$$W_i(x) = \frac{v_i}{\sum_{a_j \in S_i} \operatorname{rank}_i(a_j)}$$

$$\Phi_{\text{new}}(x) = \Phi_{\text{old}}(U(x)) + \sum_{i=1}^{k} W_i(x).$$

- A smaller denominator (fewer assigned agents or generally lower total rank) increases $W_i(x)$, reflecting a supervisor's preference for light loads.

- A set of well-ranked (highly preferred) agents yields a smaller sum of ranks, further boosting $W_i(x)$.

Global vs Local Optima in Reward Allocation

Local Optima:

- Result from best-response dynamics—no agent can unilaterally improve utility
- Driven by potential function $\Phi(U) = \sum \log(U_i)$
- Ensure stability but may be suboptimal globally

Why Global Matters:

- Local equilibrium ≠ global maximum social welfare
- Gap defined as:
- $\Delta opt = \max(SW(r)) - SW(r^*)$

Strategies Toward Global Optimality:

1. Randomized Initialization – Multiple trials to explore diverse equilibria
2. Simulated Annealing – Accept occasional worse moves to escape local traps
3. Multi-Stage Optimization – Gradual refinement toward better solutions

Future Work

1. **Decentralized Multi-Player Multi-Armed Bandit Framework**
   - **Extend the model where rewards and preference orders are unknown.**
   - **Agents learn optimal allocations over time.**
2. **Agent Communication & Coordination**
   - **Enable communication to reduce collisions and improve decision-making.**
   - **Drive the system toward a stable matching that respects agent and arm preferences.**