A Configurable Multi-Level Discount Allocation System for Performance-Based Incentives in Health Networks

Trinetra Kumar Sinha (2024pgcscs19)

Abstract

Incentive distribution in large-scale healthcare networks is a critical problem requiring fairness, transparency, and adaptability. This Assingment presents a rule-based, configurable discount allocation system designed for RedHealth, a healthcare organization managing hospitals and their agents. Our approach features a two-level allocation mechanism that first assigns discounts to hospitals based on macro-level performance metrics and then distributes hospital-level budgets to agents based on individualized KPIs. The system is driven by a configuration file that enables tuning of feature weights and allocation constraints. Results demonstrate consistent adherence to fairness principles, normalization logic, and user-defined boundaries, with support for multiple test scenarios and edge cases.

1 Introduction

Efficient and transparent distribution of incentive budgets is crucial in hierarchical organizations. In healthcare systems like RedHealth, both hospital-level outcomes and agent-level contributions must be accounted for when allocating discounts. This Assingment introduces a customizable discount allocation framework capable of fairly distributing a global incentive budget across multiple hospitals and agents using weighted feature metrics and configurable normalization.

2 Objective

The primary objective of this study is to develop a flexible, fair, and scalable system that:

- Allocates a predefined global discount kitty across hospitals based on organizational performance.
- Further distributes hospital-specific discount allocations among agents using micro-level KPIs.
- Maintains strict adherence to minimum and maximum discount constraints.
- Allows parameter configuration through a JSON file, enabling extensibility and domain adaptability.

3 System Design and Architecture

3.1 Overview

The system comprises two main components:

- `red_health_allocation.py`: The interactive driver script that collects input, computes scores, allocates discounts, and prints final output.
- `allocator.py`: A modular utility that performs agent-level allocation using normalized weighted scoring.

3.2 Configuration-Driven Design

All weights, constraints, and feature ranges are extracted from a central `config.json` file. This includes:

- Feature weights for hospitals and agents
- Min/max caps on discounts
- Normalization bounds

4 Methodology

4.1 Input Collection

The system prompts the user to enter:

- Number of hospitals
- Global discount budget
- Min/max hospital and agent-level discount constraints
- Feature values for each hospital and agent

4.2 Score Computation

Hospitals are evaluated on:

- Performance Score
- Revenue
- Customer Rating
- Service Standard

Agents are evaluated on:

- Performance metrics (e.g., performance score, targets)
- Client handling (e.g., retention rate, satisfaction)
- Work efficiency (e.g., upselling, workload complexity)

Each value is normalized using:

$$\text{Normalized Value} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

or pre-defined range-based normalization for live input.

4.3 Allocation Logic

Hospital Allocation:

$$\text{Hospital Share} = \frac{\text{Hospital Score}}{\text{Total Score}} \times \text{Global Kitty} \tag{2}$$

Agent Allocation:

$$\text{Agent Share} = \frac{\text{Agent Score}}{\text{Total Score in Hospital}} \times \text{Hospital Kitty} \tag{3}$$

Allocated amounts are clipped within configured min–max values.
A textual justification is generated for each agent.

5 Results and Evaluation

5.1 Sample Execution (See sample.txt)

Input: 3 hospitals, each with 3 agents
Global Kitty: 500,000
Hospital Allocations: Based on relative performance (e.g., H2 received 224,693.75)
Agent Allocations: Fairly distributed using agent-specific metrics
Sample output:

```
Agent H2_A1 → 106,711.63
Agent H3_A3 → 20,146.55
```

5.2 Test Cases (See test_cases.txt)

- Normal Case: Performance-proportional allocation confirmed
- All-Same Scores: Equal distribution validated
- Rounding Edge Case: Single agent received total kitty with rounding correctness

5.3 JSON Output

The system provides structured JSON for downstream processing or dashboard integration:

```
{
  "summary": "Total kitty 500000.0 distributed to hospitals and their agents.",
  "allocations": [...]
}
```

6 Discussion

This system ensures:

- Fairness: Through normalized scoring and weight-based proportional allocation
- Transparency: Detailed justifications for each allocation
- Scalability: Supports any number of hospitals and agents
- Modularity: Easily integrates with APIs, web apps, or dashboards

6.1 Limitations

- Relies on manual input for live mode
- Does not adapt dynamically to temporal changes in performance over time

7 Conclusion

The proposed allocation system for RedHealth successfully balances fairness, configurability, and clarity. Through normalization, rule-based weight scoring, and bounded allocations, it ensures incentives are justifiably distributed across hierarchies. The modular structure and JSON-driven configuration enable extensibility and domain transferability to other enterprise settings.

8 Future Work

- Integration with real-time databases or dashboards
- Time-series analysis of agent performance trends
- Hybrid AI + rule-based allocation (e.g., reinforcement learning)
- Visualization modules for allocation heatmaps

References

- Internal design files: `allocator.py`, `red_health_allocation.py`
- Configuration schema: `config.json`
- Test scenarios: `test_cases.txt`
- Execution logs: `sample.txt`