AI Scheduler Using Deep Reinforcement Learning

School of Data Science & Artificial Intelligence
Indian Institute of Technology Guwahati

Team HIVE

Aditya Sunil Lambat (230150002)

Asif Nazeer Hossain (230150004)

Yuvraj Nim (230150030)

April 8, 2025

Contents

1 Introduction

This report presents our AI-based task scheduling system that employs deep reinforcement learning to optimize resource allocation in professional environments. The system integrates natural language processing (NLP), simulation modeling, and a Deep Q-Network (DQN) agent to make informed decisions about task assignments by considering multiple factors simultaneously: employee skill profiles, workload distribution, task priorities, deadlines, and resource constraints.

Traditional scheduling approaches rely on static rules or manual intervention, which leads to suboptimal resource utilization and missed deadlines. By leveraging reinforcement learning techniques, our system adapts to complex workplace environments while continuously improving its decision-making capabilities through experience.

2 System Architecture

Our AI scheduler consists of three primary components:

1. Natural Language Processing (TaskNLP) module
2. Task Environment Simulation
3. Deep Q-Network Scheduler Agent

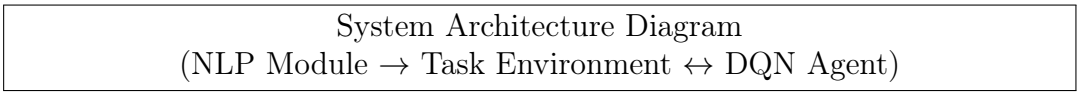

Figure 1: System Architecture Overview

2.1 Natural Language Processing (TaskNLP)

The TaskNLP module serves as the interface between human task descriptions and the computational scheduling system. It utilizes spaCy's "en_core_web_lg" model to perform text analysis on task descriptions, extracting essential attributes such as type, priority, complexity, duration, deadline, and required skills.

For example, the input "We urgently need advanced cloud infrastructure setup with basic documentation skills" would be processed to produce structured data:

```
{
  "task_type": "Infrastructure",
  "skill_requirements": [0.19, 0.18, 0.27, 0.38, 0.9, 0.8, 0.3, 0.15],
  "priority": 5,
  "duration": 4.1,
  "deadline": 7,
  "complexity": 5
}
```

2.2 Task Environment Simulation

The simulation creates a realistic work environment with diverse employee profiles and dynamic task generation. Employees are modeled with multi-dimensional skill vectors (specialists vs. generalists), varying availability patterns, fatigue modeling, and different seniority levels. Tasks are generated with varying priorities (1-5), deadlines, complexity scores, skill requirements, and estimated durations.

2.2.1 Employee Modeling

The simulation accounts for:

- **Skill Profiles**: Multi-dimensional skill vectors representing proficiency levels
- **Fatigue Modeling**: Affects task completion efficiency over time
- **Availability Patterns**: Different work schedules and capacity constraints

The fatigue function is modeled as:

$$F(e,t) = \alpha \cdot C_e(t) + \beta \cdot \sum_{i=1}^{n} w_i \cdot T_i \tag{1}$$

Where $C_e(t)$ represents current workload and T_i represents task complexity.

2.3 Deep Q-Network Scheduler

The DQNSchedulerAgent learns to make optimal task assignments through interaction with the simulated environment. It uses a neural network to approximate the Q-function, predicting expected future rewards for different actions given the current state.

2.3.1 Network Architecture

Our implementation features:

- Input layer: Accepts combined state vector
- Hidden layers: Three fully-connected layers with 128, 256, and 128 neurons
- ReLU activation functions for hidden layers
- Linear activation for the output layer

2.3.2 Key Components

- **Experience Replay**: Stores previous experiences to improve learning stability
- **Double DQN**: Uses separate networks for action selection and evaluation
- **Exploration Strategy**: Employs epsilon-greedy policy that gradually transitions from exploration to exploitation

2.3.3 State and Action Spaces

The state vector (28 dimensions) combines task attributes, employee attributes, and environmental factors. For each task, the agent can assign it to any available employee or reject/postpone it.

2.3.4 Reward Function

The reward function encourages optimal assignments:

$$R(s,a) = w_p \cdot P + w_s \cdot S - w_f \cdot F - w_d \cdot D - w_o \cdot O \tag{2}$$

Where P is task priority, S is skill match score, F is fatigue penalty, D is deadline urgency, O is workload imbalance, and w values are weighting coefficients. Task rejection incurs a negative reward proportional to the task's priority.

3 Training Process

3.1 Training Protocol

The agent begins with random weights and follows this procedure:

1. Initialize the DQN agent and task environment
2. For each episode:
   (a) Reset the environment (new employees and initial task queue)
   (b) For each simulated day:
      i. Generate new tasks
      ii. For each task: observe state, select action, execute action, store experience
      iii. Sample batch from replay memory and update network weights
      iv. Update target network every 100 steps
   (c) Evaluate performance metrics
3. Save trained model and performance data

Each episode simulated 30 working days with a team of 10 employees handling approximately 150-200 tasks per episode. We trained the model for 100 episodes on an NVIDIA RTX 3080 GPU.

3.2 Hyperparameter Configuration

Parameter	**Value**
Learning rate	0.001
Discount factor (γ)	0.95
Initial exploration rate (ϵ)	1.0
Final exploration rate	0.01
Exploration decay rate	0.995
Replay memory size	10,000
Batch size	64
Target network update frequency	100 steps

Table 1: DQN Hyperparameters

4 Results and Analysis

4.1 Performance Comparison

We compared our DQN scheduler against three baseline methods: Random Assignment, Greedy Skill Matching, and Rule-Based Priority scheduling.

Figure 2: Performance Comparison of Scheduling Methods

The DQN scheduler demonstrated significant improvements:

- **Task Completion**: Overall task completion rate increased from 78% (best baseline) to 92% (DQN)

- **Priority Handling**: High-priority task completion rate reached 97%, compared to 85% for the rule-based system
- **Assignment Speed**: Average time to assignment decreased by 43%
- **Workload Balance**: Standard deviation of workload across employees decreased by 37%
- **Skill Utilization**: Average skill match score of 0.82 compared to 0.71 for the greedy approach
- **Employee Wellness**: Fatigue levels remained optimal for 89% of the workforce

4.2 Adaptation to Varying Conditions

The scheduler maintained 84% completion rate even when task volume doubled (compared to 62% for rule-based system) and adapted well to skill shortages and deadline pressures.

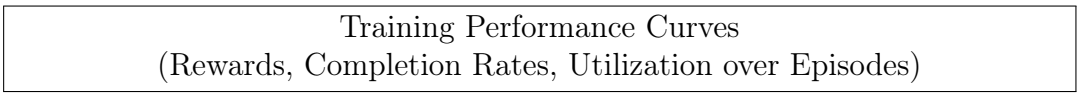

Figure 3: Performance Metrics During Training

4.3 Ablation Studies

To understand component contributions:

- Removing the NLP module reduced task completion rates by 14%
- Using standard DQN instead of Double DQN led to 8% lower performance
- Simplifying the employee fatigue model decreased productivity by 23%
- Removing workload balance from the reward function decreased team performance by 11%

5 Conclusions and Future Work

5.1 Summary of Contributions

Our AI scheduler system successfully demonstrates the application of deep reinforcement learning to task assignment in professional environments. Key contributions include integrating NLP capabilities, developing a realistic simulation environment, implementing a Double DQN architecture, and showing significant improvements over traditional methods.

5.2 Limitations

Despite promising performance, limitations include:

- NLP module sometimes misinterprets complex task descriptions
- Substantial training data requirements
- Need for retraining when adapting to new task types
- Simulation may not capture all nuances of workplace dynamics
- Current implementation doesn't handle collaborative tasks

5.3 Future Directions

Future research directions include:

- Multi-agent reinforcement learning for collaborative tasks
- Models accounting for employee skill development over time
- Hierarchical reinforcement learning for strategic and tactical decisions
- Transformer-based architectures for improved language understanding
- Real-world validation studies
- Handling uncertainty in task duration estimates
- Interactive interfaces for human-AI collaboration

The system has potential applications across industries including IT service management, customer support, healthcare staff scheduling, manufacturing, and project management. By optimizing resource allocation, such systems can improve productivity, reduce costs, and enhance employee satisfaction through more equitable workload distribution, particularly valuable in dynamic work environments.