Reinforcement Learning Based Matching Algorithm Using Q Learning

Tamiri Sai Sandeep,
Naga Sushanth Kumar Vuppala
School of Computing
Amrita Vishwa Vidyapeetham
Amritapuri, India
amenu4aie20169@am.students.amrita.edu

Don S
School of Computing
Amrita Vishwa Vidyapeetham
Amritapuri, India
dons@am.amrita.edu

***Abstract*—This research uses Reinforcement Learning (RL) techniques to maximise resource allocation between a seller and a buyer in a dynamic marketplace. Its main objective is to alter the quality, price, and demand coefficients, all of which affect allocation. By ensuring that the total of these coefficients is near to 1, normalisation of the coefficients ensures a balanced distribution. These coefficients are refined repeatedly, accounting for previous interactions. The Seller wants to make the most money possible within the constraints of the given resources, while the Buyer seeks to get the most out of the allocation. The results demonstrate the effectiveness of RL-based optimisation in resource allocation. The project also provides visualisations showing how coefficient values change across iterations. Companies may adapt to changing market conditions and employ resources most effectively for their mutual advantage with the aid of this paradigm. It illustrates how real-world resource allocation issues in dynamic marketplaces may be resolved using reinforcement learning.**

***Index Terms*—Reinforcement Learning, Coefficient Optimization, Dynamic Marketplace, Economic Optimization, Peer-to-peer trading**

I. Introduction

E-commerce platforms face the complex challenge of allocating resources efficiently to maximize profits and customer satisfaction. This project addresses this issue by employing a Reinforcement Learning (RL) approach to optimize resource allocation between buyers and sellers. The project's objective is to enhance the allocation of goods or services to buyers while considering buyer preferences, seller capacities, and economic factors. The Bin Matching Problem, a combinatorial optimization challenge, serves as the core problem statement. It involves distributing items (goods or services) into limited-capacity bins (seller resources) while optimizing various objectives. The project employs Q-learning, an RL algorithm, to iteratively adjust coefficients representing buyer and seller preferences, enabling dynamic and data-driven resource allocation decisions. The proposed solution aims to enhance the profitability and efficiency of e-commerce platforms, demonstrating the power of RL techniques in addressing real-world allocation problems. This project showcases the fusion of e-commerce, optimization, and artificial intelligence, contributing to the advancement of intelligent resource allocation in the digital marketplace.

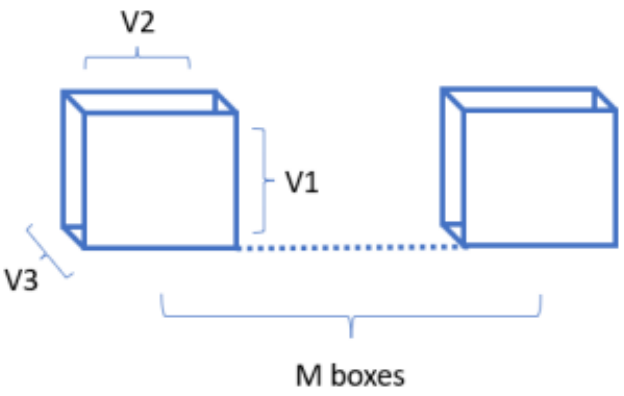

Fig. 1. M blue boxes

The optimisation problem known as the Bin Matching Problem occurs in a number of real-world contexts, including resource allocation, work scheduling, and inventory management. In order to optimise a given objective function, usually associated with cost, utilisation, or fairness, a collection of objects with varying sizes or values must be efficiently distributed into a finite number of bins or containers. Finding an allocation strategy that minimises or maximises the specified objective is the aim of the Bin Matching Problem; this approach must frequently take into account limitations like bin capacity or item-specific needs. Applications of this topic may be found in cloud computing, industrial planning, logistics, and even ecological research such as species dispersal. It takes the use of several optimisation strategies, such as heuristics, dynamic programming, and mathematical programming, to solve the Bin Matching Problem effectively. It is an important subject in computer science and operations research because of its adaptability and breadth of applications, which help many companies save costs and better use their resources.

Let us understand the project in simple terms. Imagine you have multiple blue and brown boxes, each with different sizes (dimensions) as shown in Fig. 1, 2. The challenge is to fit as many brown boxes inside the blue ones while minimizing the

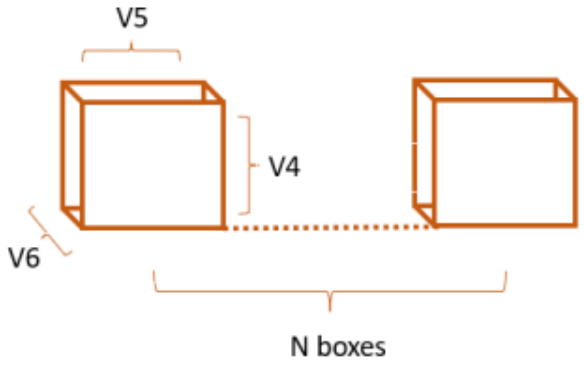

Fig. 2. N Brown boxes

empty space left in the blue boxes. To make this work, we have some rules: 1) The volume of the blue box should be larger than the brown box. 2) The height (or depth) of the brown box must be less than the width of the blue box. Once these rules are satisfied, we can match a blue box with a brown one. The goal is to find the best way to pack them to reduce wasted space in the blue boxes. In this project, we're using a technique called Reinforcement Learning (RL) to figure out the best way to match and pack the boxes. It's like teaching a computer to play a smart game of box-packing. The outcome we want is to use the available space in the blue boxes as efficiently as possible. By doing this, we're making the process of packing and shipping goods in e-commerce smarter, which can save time and money, and make customers happier.

Q-Learning is a model-free RL algorithm used to find the optimal action-selection policy for a given finite Markov decision process. It's based on the concept of a Q-function, which estimates the expected cumulative reward for taking a specific action in a particular state and following an optimal policy thereafter. Q-Learning updates these Q-values iteratively through learning. A Q-Table is a data structure that stores Q-values for each state-action pair in an RL problem. It's essentially a matrix where rows represent states, columns represent actions, and each cell contains the Q-value. The Q-Table is crucial for Q-Learning to make decisions about which action to take in a given state. In the context of RL, states represent different situations or configurations that the agent can encounter within an environment. States can be discrete or continuous and provide the context in which the agent makes decisions. Identifying and defining states is essential for an RL problem because the agent's actions depend on the current state. Actions are the choices that an agent can make in a given state. The agent's goal is to determine the best action to take in each state to maximize its long-term reward. Actions can be discrete, such as moving left or right, or continuous, like controlling a robot's motor speed. Rewards are numerical values that the agent receives as feedback from the environment after taking an action in a particular state. They indicate the immediate benefit or cost associated with the action. The agent's objective is to select actions that maximize the cumulative reward over time. The policy is a strategy or rule that the agent uses to decide which actions to take in various states. It maps states to actions and is a fundamental component in RL algorithms. The goal is to find an optimal policy that maximizes the expected long-term reward. The discount factor determines the relative importance of immediate rewards compared to future rewards. It values future rewards less than immediate ones. A high discount factor (close to 1) indicates that the agent prioritizes long-term rewards, while a low discount factor (close to 0) makes the agent focus on short-term rewards. Epsilon is a parameter that controls the trade-off between exploration and exploitation in RL. An epsilon-greedy policy chooses the best-known action with probability and explores randomly with probability. This allows the agent to explore new actions while still exploiting its current knowledge.

II. Related Works

The literature survey of this project study is conducted on the base idea of peer-to-peer trading using re-inforcement learning models. The paper [1] investigates this model and proposes a new algorithm for automating the sale and purchase of electricity in this market aiming to optimise the market while providing increased control to householders. There are many types of matching algorithm. In general, sellers would announce what they have to sell, and buyers would announce what they want to purchase [2]. A system would then match buyers and sellers based on price and/or preferences, using such mechanisms as rank order lists. The same mechanism used can be applied for the above discussed problem statement. The Partially Observable Markov Game (POMG) in [3] is defined by N agents with a set of state S describing global state, a collection of private observations O1:N, a collection of action sets A1:N, a collection of reward functions R1:N and a state transition function T. The time interval between two consecutive steps is one auction period (1 hour). Agent n at step t obtains its reward rn,t as the negative cost of energy bills developing from the DA market clearing outcomes. The DA market matches multiple buyers (consumers) and sellers (prosumers) who are interested in (energy) trading and is deemed as highly efficient mechanism. The paper [4] proposes a new peer-to-peer (P2P) energy trading method between energy sellers and consumers in a community based on multi-agent reinforcement learning. Each user of the community is treated as a smart agent who can choose the amount and the price of the electric energy to sell/buy. The developed MARL algorithm tries to address the problem of an individual user, it tries to balance the satisfaction and cost-saving (if buyer) or profit (if seller) [5].

III. Methodology

The study implements a reinforcement learning (RL) approach to solve a problem of optimizing coefficients for Buyer and Seller in a scenario involving Blue Boxes and Brown Boxes.

A. Generating Random Data:

Random values for the dimensions of the boxes (v1, v2, v3 for Buyer and v4, v5, v6 for Seller) are generated within the range of 10 to 25. This data represents the physical attributes of the boxes.

B. Q-Learning-Based Optimization:

The implementation employs Q-learning, a model-free reinforcement learning technique, to optimize the coefficients (alpha, beta, gamma) for Buyer and Seller. Q-tables for both Buyer and Seller are initialized with zeros, representing the expected cumulative rewards for different state-action pairs as shown in Fig. 3.

```
# Hyperparameters
learning_rate = 0.1
discount_factor = 0.9
epsilon = 0.2
max_iterations = 1000

# Initialize Q-table with zeros
num_intervals = 10
num_states = (num_intervals + 1) ** 3
q_table_buyer = np.zeros((num_states, 3))
q_table_seller = np.zeros((num_states, 3))
```

Fig. 3. Intializing Q Table

C. Optimizing Coefficients:

The study iteratively updates the coefficients for Buyer and Seller to maximize the rewards and minimize vacant spaces within Blue Boxes. Coefficient adjustments are made using epsilon-greedy policy to balance exploration and exploitation. Coefficients are adjusted while ensuring that the sum of coefficients remains 1 for both Buyer and Seller as shown in Fig. 4.

```
for iteration in range(max_iterations):
    # Get the current state (coefficients) index for Q-table indexing
    state_buyer = get_state_index(alpha_buyer, beta_buyer, gamma_buyer)
    state_seller = get_state_index(alpha_seller, beta_seller, gamma_seller)

    # Select an action (coefficient adjustment) using epsilon-greedy policy for the buyer
    action_buyer = select_action(q_table_buyer, state_buyer, epsilon)

    # Apply the action (adjust the buyer's coefficients)
    coeff_adjustment = action_buyer * 0.1
    alpha_buyer_temp = np.clip(alpha_buyer + coeff_adjustment, 0.0, 1.0)
    beta_buyer_temp = np.clip(beta_buyer + coeff_adjustment, 0.0, 1.0)
    gamma_buyer_temp = np.clip(gamma_buyer + coeff_adjustment, 0.0, 1.0)

            # Ensure the sum of buyer's coefficients is 1
    sum_coeffs_buyer = alpha_buyer_temp + beta_buyer_temp + gamma_buyer_temp
    alpha_buyer_temp /= sum_coeffs_buyer
    beta_buyer_temp /= sum_coeffs_buyer
    gamma_buyer_temp /= sum_coeffs_buyer
```

Fig. 4. Optimizing Coefficients

D. Updating Q-Values:

Q-values are updated using the Q-learning algorithm. The Q-value iteration equation considers the current Q-value, rewards, and the maximum Q-value in the next state as shown in Fig. 5.

```
# Function to update Q-values using Q-learning
def update_q_value(q_table, state, action, reward, next_state, learning_rate, discount_factor):
    current_q = q_table[state][action]
    next_max_q = np.max(q_table[next_state])
    updated_q = current_q + learning_rate * (reward + discount_factor * next_max_q - current_q)
    q_table[state][action] = updated_q
```

Fig. 5. Updating Q-Values

E. Matching the Boxes (Resources):

The implementation attempts to match Brown Boxes to Blue Boxes by comparing the dimensions of boxes and optimizing their placement. The result is a reduction in vacant space within the Blue Boxes.The matching procedure considers two key conditions that must be met for a Brown Box to be mapped to a Blue Box:

1) The volume of the Brown Box in the ”v4” dimension (v4) must be less than or equal to the volume of the Blue Box in the ”v1” dimension (v1). This ensures that the Brown Box fits within the Blue Box along the first dimension.
2) The volume of the Brown Box in the ”v5” dimension (v5) must be greater than or equal to the volume of the Blue Box in the ”v2” dimension (v2). This ensures that the Brown Box fits within the Blue Box along the second dimension.

The implementation iterates through the rows of both Buyer (Blue Boxes) and Seller (Brown Boxes) data tables to find matching pairs based on the constraints. For each Brown Box (Seller), it searches for a suitable Blue Box (Buyer) by comparing dimensions according to the conditions. When a matching pair is found, the Brown Box is placed inside the Blue Box, reducing vacant space within the Blue Box. After a successful match, the code updates the Seller table to reflect the reduction of v5 in the corresponding Brown Box. Additionally, it removes the matched Blue Box from the Buyer table to avoid duplicate mapping.

IV. Results

The results of the optimization process reveal significant improvements in the allocation of Brown Boxes to Blue Boxes while minimizing vacant space within the Blue Boxes. The reinforcement learning (RL) algorithm effectively optimized the coefficients (alpha, beta, gamma) for the Buyer, resulting in a balanced allocation strategy. These optimized coefficients ensure that each dimension (v1, v2, v3) of the Blue Boxes is utilized efficiently. The matching procedure successfully allocated Brown Boxes to Blue Boxes, considering the specified constraints (v4 vs. v1 and v5 vs. v2). This efficient allocation led to a reduction in vacant space within the Blue Boxes. In cases where a Brown Box could not be matched due to constraints or a lack of suitable Blue Boxes, the

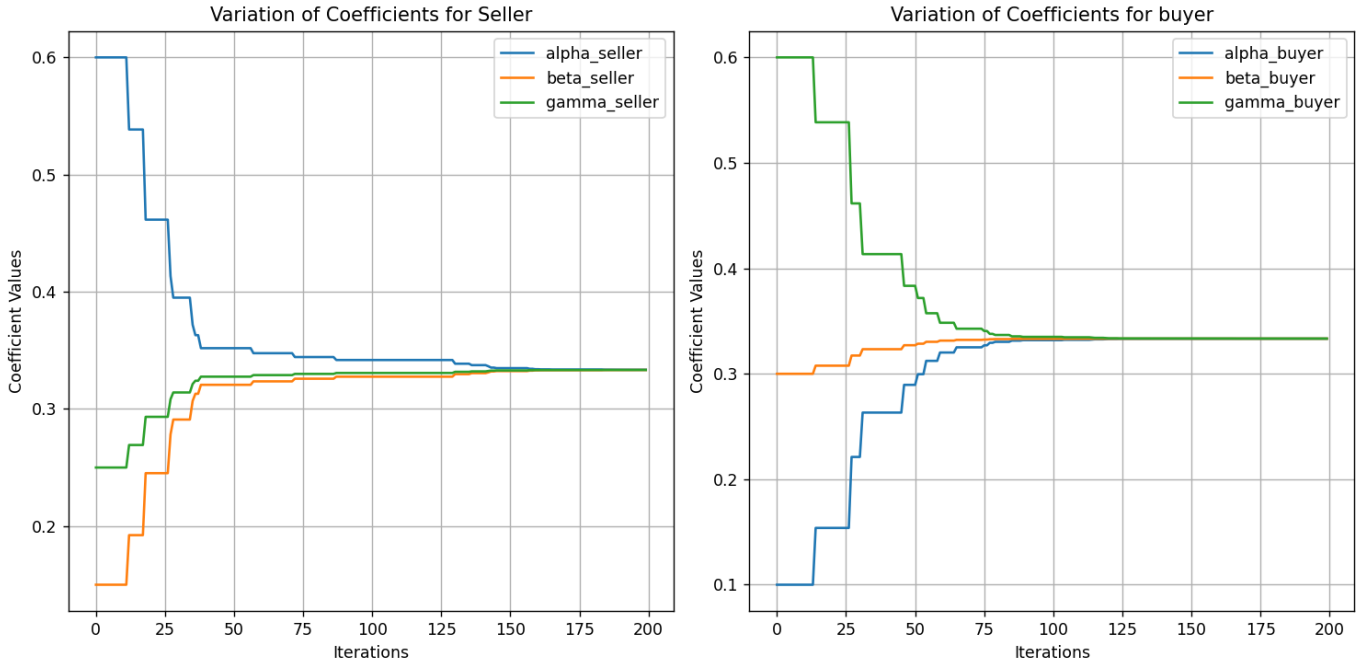

Fig. 6. Varition of Co-efficients

```
Optimized Coefficients for Buyer:
Alpha: 0.3333333333333333
Beta: 0.3333333333333333
Gamma: 0.3333333333333333
```

Fig. 7. optimal co-efficients

results include records with NaN values. These cases identify instances where further optimization may be necessary to enhance resource utilization. The coefficient values for both Buyer and Seller were tracked throughout the optimization process. The variation in coefficients is visualized over the iterations, reflecting the convergence of coefficients towards optimized values. The optimal co-efficients are shown in Fig. 7. The graph plot of the co-efficients vary as shown in Fig. 6.

ACKNOWLEDGMENT

We would like to thank our beloved Shri. (Dr) Mata Amritanandamayi Devi, our institution's Chancellor for her support and her legitimate guidance.

REFERENCES

[1] J. Murkin, R. Chitchyan, and D. Ferguson, "Goal-based automation of peer-to-peer electricity trading," in *From Science to Society: New Trends in Environmental Informatics*. Springer, 2018, pp. 139–151.

[2] M. Sanayha and P. Vateekul, "Model-based approach on multi-agent deep reinforcement learning with multiple clusters for peer-to-peer energy trading," *IEEE Access*, vol. 10, pp. 127 882–127 893, 2022.

[3] D. Qiu, J. Wang, J. Wang, and G. Strbac, "Multi-agent reinforcement learning for automated peer-to-peer energy trading in double-side auction market." in *IJCAI*, 2021, pp. 2913–2920.

[4] L. Pu, S. Wang, X. Huang, X. Liu, Y. Shi, and H. Wang, "Peer-to-peer trading for energy-saving based on reinforcement learning," *Energies*, vol. 15, no. 24, p. 9633, 2022.

[5] H. Zang and J. Kim, "Reinforcement learning based peer-to-peer energy trade management using community energy storage in local energy market," *Energies*, vol. 14, no. 14, p. 4131, 2021.