# RL-Driven Warehouse Item Relocation Optimization

## Problem Statement:

Modern warehouses face a constant challenge of optimizing item placement to balance storage density, retrieval efficiency, and operational cost. Traditional manual or rule-based slotting methods struggle to adapt to dynamic inventory profiles and evolving demand patterns.

This project aims to develop an RL-based optimization system capable of determining the best item relocation strategies — maximizing space utilization while minimizing unnecessary movement operations. The system will explore multiple relocation sequences within static warehouse configurations to improve storage efficiency and accessibility, with potential extension to simulate ongoing warehouse operations.

## Related Work

1. **"A Q-learning-based algorithm for the block relocation problem" — Liu et al., 2025**
   Method: Q-learning to minimize relocations in block stacking systems.
   Link: https://link.springer.com/article/10.1007/s10732-024-09545-y
   Difference: While Liu et al. focus on minimizing relocations in isolated block-stacking scenarios, our project aims to develop a general RL framework for optimizing relocation across a full warehouse layout, balancing density and movement cost across multiple item types and zones.
2. **"Deep Reinforcement Learning for Task Assignment and Shelf Reallocation in Smart Warehouses" — 2025**
   Method: Deep RL for task assignment and shelf reallocation to improve operational efficiency.
   Link:https://www.researchgate.net/publication/380049934_Deep_Reinforcement_Learning_for_Task_Assignment_and_Shelf_Reallocation_in_Smart_Warehouses
   Difference: This work focuses on dynamic shelf reallocation and task assignment for operational throughput, whereas our approach emphasizes static warehouse densification and item relocation planning, generating efficient layouts prior to peak operations.
3. **Arslan, E. (2025). Optimizing Human-Centric Warehouse Operations: A Digital Twin Approach Using Dynamic Algorithms and AI/ML.**
   Link: https://doi.org/10.51551/verimlilik.1524701
   Method: combining Digital Twin technology with advanced AI/ML analytics to dynamically adjust operational strategies based on real-time data collected from warehouse activities.

## Expected Outcomes:

- A **functional prototype (MVP)** capable of generating optimized warehouse configurations based on defined parameters and constraints.
- A **synthetic dataset** modeled after realistic warehouse conditions, ensuring data privacy and compliance while maintaining validity.
- A **visual interface or dashboard** for analyzing warehouse states and tracking relocation improvements.
- A **Feedback Loop** where the results of the optimized parameters are used as input for the model, starting an iterative process.
- A **documented framework** that can later be extended or integrated with real warehouse data for production-level deployment.

## Tech Stack and Tools:

- Programming Language: **Python**
- Simulation: **SimPy**
- Optimization & AI: **PyTorch (Reinforcement Learning), TensorFlow, custom heuristic solvers**
- Data Handling: **NumPy, Pandas**
- Visualization: **Matplotlib, Plotly, or a web-based 2D warehouse viewer (e.g., Dash, Streamlit, or React front-end)**

## Challenges:

- **Limited Access to Real-World Data:** Due to data confidentiality, direct use of warehouse layout and operational datasets is not possible. However, the structure and characteristics of those files will be used as a reference framework to generate a realistic synthetic dataset. This approach will ensure that the simulated warehouse data maintains real-world relevance while complying with privacy and ownership restrictions.
- **Combinatorial Complexity**: The number of possible item arrangements grows exponentially, making exhaustive search infeasible.
- **Trade-off Tuning:** Balancing competing objectives (e.g., storage density vs. movement cost) demands careful design of reward or fitness functions.
- **Generalization:** Ensuring that the learned or evolved relocation strategies perform well across varying warehouse configurations and demand patterns.

## Methodology:

**Step 1: Warehouse Representation**

- Represent the warehouse as a structured data grid (e.g., a 2D NumPy array or Pandas DataFrame), where each cell corresponds to a storage location.
- Encode item information such as SKU type, size, access frequency, and zone.
- Define constraints such as shelf capacity, forbidden zones, or adjacency rules.
- Use assumptions and educated guesses based on the bibliography and scientific sources.

**Step 2: Synthetic Data**

- Obtain a dataset from the simulated process in the warehouse according to the parameters chosen.
- Use this data to feed machine learning models, used for prediction and prescriptive actions in the simulated warehouse.

**Step 3: Optimization Engine**

- Reinforcement Learning (PyTorch): Train an RL agent where actions = item movements and rewards = improved density minus movement cost.
- Heuristic Search (optional): Greedy or simulated annealing methods may be used to generate initial feasible layouts for the RL agent.

**Step 4: Fitness / Objective Function**

- Define a composite objective to guide optimization:

$$\text{Fitness} = \alpha \times \text{Storage Density} - \beta \times \text{Movement Cost}$$

  where α and β weight the relative importance of density versus movement.
- Evaluate layouts on metrics such as total occupied space, number of relocation steps, and accessibility of frequently used items.

**Step 5: Reproducibility**

- GitHub repo with codes, respective documentation of the study and manuals of use.
- Scientific result with methodology and discussion.