TEAM – NEURAL NINJAS

PARTICIPANTS-

1. RICHA VERMA - 25mcss02@iiitdmj.ac.in (9981579963)
2. ATUL KUMAR GUPTA - 25mcss06@iiitdmj.ac.in (7723807525)
3. SHASHANK SIROTHIYA - 25mcsa19@iiitdmj.ac.in (8269884717)

Warehouse Picklist Optimization Engine

1. Problem Understanding:

We need to design an algorithm that creates and schedules picklists to maximize SKU fulfillment before the loading deadlines in Quick-commerce warehouses.

Orders come from multiple stores and include multiple SKUs. These orders can be split across different picklists. A single picklist can include SKUs from multiple orders, provided it meets certain constraints such as **weight limit**, **quantity limit**, and **maximum number of shops per picklist**.

A picklist (or even part of it) adds value only if it is completed before the order cutoff time. Partial fulfillment is still valuable, but any items completed after the cutoff are considered wasted effort.

Additional constraints:

- A picklist can only include items from **one zone**.
- A picker can work on **one picklist at a time**.
- A picker can only work on a picklist if the entire picklist fits within their shift.

2. Assumptions

We have made the following assumptions in designing the algorithm:

1. Fragile Handling

    • Fragile SKUs are stored only in dedicated fragile zone: "FRAGILE_FD"

2. Time Estimation

    • Travel, bin switching, unit picking are the same for each SKU.

    • Unloading and staging time is the same for all orders, i.e., independent of quantities present in the order.

3. Priority Handling

    • Higher-priority orders have tighter deadlines.

4. Picker Behavior

    • Every picker works with the same efficiency.

    • Picker is available for picking or is busy with picking. (No intervention during shift timing)

5. Distance Estimation

    • Lower Bin rank means close to the entrance of the zone

    • Lower Floor, lower aisle and lower rack mean closer to entrance to the zone

    • No congestion effects due to multiple pickers in same zone

6. SKU Property

    • We have infinite instances of each SKU like unbounded knapsack

    • Ignoring dimensions of the SKUs.

    • Fixed pickup time for each SKU.

3. Algorithm

It's a rule-based decision-making algorithm.

Step 1: Zone-Wise Decomposition

Orders are first **partitioned by the warehouse zone.** Since picklists can be from a single zone only, we are grouping orders based on zone.

Now for each Zone following steps are made:

Step 2: Candidate Item Scoring (ATC-Based)

For every SKU:

1. Compute an **ATC (Apparent Tardiness Cost) score**, capturing:
    a. Pick Density
    b. Urgency (deadline proximity)

$$\textit{Pick Density} = (sku\_quantity) \;/\; (processing\_time)$$
$$\textit{Process time} = bin\_to\_bin\_time \; + (sku\_quantity * time\_to\_pick\_per\_unit)$$

While calculating process time, we take one bin movement assuming that we need to move to a different bin to pick up this item.
Example:
Let's take 1 bin movement = 30 seconds
For Item A - sku_quantity = 1
So, Pick Density = $\frac{1}{(30+1*5)}$ = 0.028
For Item B - sku_quantity = 4
So, Pick Density = $\frac{4}{(30+4*5)}$ = 0.08
Pick Density will be more for Item B that has a higher quantity.

2. Slack = time_until_cutoff - process_time – overhead
   Where overhead = start to zone time + zone to staging area time.
   Slack (normalized) = $\frac{Slack}{time\_until\_cutoff}$
   Here all the time values are taken in seconds to maintain relative comparison.

3. Urgency: math.exp(exponent)
   Where exponent = $\frac{-\max(slack,0)}{ATC_K}$
   Here ATC_K is a constant used to introduce panic factor, i.e. if this is smaller value (less than1), urgency will contribute more to ATC_Score i.e. our algorithm will prioritize urgent i.e. early cutoff items and if this value is larger (more than 1) algorithm will favor large sku quantities items. For this problem, we have taken ATC_K as 2.0 to maximize throughput.

   ATC_SCORE = PICK DENSITY * URGENCY

$$\text{ATC Score} = \underbrace{\left(\frac{\text{Quantity}}{\text{Process Time}}\right)}_{\text{Pick Density}} \times \underbrace{\exp\left(-\frac{\max(\text{Slack}, 0)}{ATC_K \times \text{Time Until Cutoff}}\right)}_{\text{Urgency}}$$

Step 4: Sorting

Candidates are sorted using the following priority order:

1. **Descending ATC score** (primary throughput and urgency driver)
2. **Order completion flag** (secondary objective)
3. Location proximity heuristics (floor, aisle, rack, bin rank)

Step 5: Seed Selection

The highest-ranked candidate becomes the **seed item** of the picklist.

Seed quantity is determined by:

- Remaining SKU quantity
- Maximum picklist unit limit
- Zone-specific weight constraints

Step 6: Picklist Growth (Greedy Expansion)

The picklist is expanded greedily by iterating through remaining candidates:

Each candidate is added only if **all constraints are satisfied**:

- Total weight ≤ zone limit
- Total units ≤ picklist limit
- Store (pod) count ≤ allowed maximum
- Picklist completion time ≤ earliest cutoff among items

A **time feasibility check** ensures that the entire picklist can be completed before its tightest deadline.

Step 7: Picklist Finalization

Once no more items can be safely added:

- Estimate final pick duration
- Assign earliest cutoff as the picklist deadline

The algorithm then repeats until all quantities in the zone are added in picklists.

Step 7: Picker Assignment

- Picklists are assigned to pickers using a greedy earliest-availability scheduling algorithm.
- All pickers across shifts are maintained in a min-heap ordered by the next available time.
- Each picklist is assigned to the earliest available picker if it can be completed within shift and deadline constraints. When a picklist exceeds remaining shift capacity, it is

safely truncated, and the remainder is requeued at the next index (i.e. i+1) to preserve order.

- This strategy guarantees zero wasted picking effort and high picker utilization while maintaining scalability.

Key Insights and Tradeoffs

1. Throughput Maximization vs Order Completion

Insight:
Maximizing SKU throughput under strict picker capacity and deadline constraints leads to partial order fulfillment.

Tradeoff:
Prioritizing order completion would require allocating picker time to low quantity SKUs, reducing total units picked. The algorithm intentionally avoids this tradeoff, treating order completion as a secondary rather than a primary objective.

2. Feasibility Enforcement vs Demand Coverage

Insight:
Strict enforcement of cutoffs and shift boundaries ensures that all assigned picklists are completed on time, resulting in zero wasted picking effort.

Tradeoff:
Some demand remains unfulfilled because infeasible picklists are intentionally not started. Relaxing feasibility constraints could increase total units picked but would introduce late work and operational risk. The algorithm here prioritizes avoiding late work.

2. Zero Wasted Effort vs Opportunistic Completion

Insight:
Avoiding late picklists ensures that all picker effort contributes to usable output.

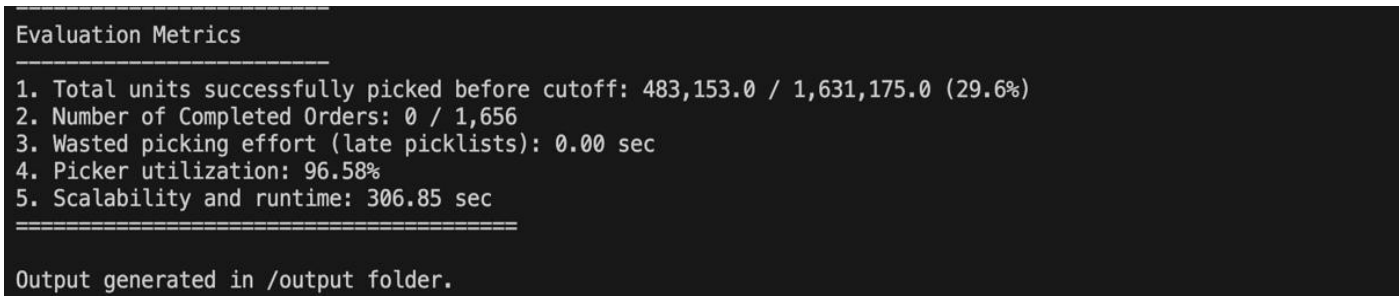
Tradeoff:
Opportunistic late or partial work could slightly increase completion rates but would introduce ambiguity in execution. The algorithm here prioritizes zero wasted effort.

OUTPUT SNAPSHOTS:

Output of the algorithm for first n-rows = 100k

1. Total units successfully picked before cutoff: 344,818.0 / 362,696.0 (95.1%)
2. Number of Completed Orders: 167 / 1,577
3. Wasted picking effort (late picklists): 0.00 sec
4. Picker utilization: 96.59%
5. Scalability and runtime: 14.30 sec

Output on entire dataset:



Despite achieving very high picker utilization, the algorithm prioritized SKU throughput under strict deadline constraints, which led to aggressive partial picking across many orders. As a result, order completion dropped at scale because finishing low-quantity order tails would have reduced overall throughput and violated the primary optimization objective.