Warehouse-Level Picklist Optimization for High-Throughput Fulfillment

TEAM GETALIFE(); - Dhairya Pandya, Rounak Agarwal, Shantanu Kharwar

GitHub Repository of the Project

Problem Understanding & Assumptions

Problem Understanding

The core objective is to design a Warehouse Picklist Optimization Engine that maximizes the number of SKU units picked and loaded before specific cut-off times. **We fundamentally understand this operation to be geared towards standard e-commerce fulfillment timelines (e.g., 1-2 day delivery), rather than a hyper-local quick-commerce model (e.g., 10-minute delivery). This critical distinction allows for strategic batching and longer optimization horizons that would not be feasible in instant-delivery scenarios.**

Consequently, this is not merely a bin-packing problem; it is a complex resource scheduling challenge involving:

- **Hard Constraints:** Strict loading deadlines (cutoffs), picker shift availability, and physical limits (weight/volume).
- **Operational Efficiency:** Minimizing non-productive time (traveling to zones, staging areas) to maximize "picking" time.
- **Fulfillment Value:** The goal is total units fulfilled; partial orders generate value, while late items generate zero value.

Assumptions

To convert this real-world scenario into a solvable algorithm, we made the following assumptions based on the dataset and operational logic:

1. **Data Relevance:** The physical dimensions (Length, Width, Height) of SKUs are of negligible importance for the picking process, as the primary constraint is Weight (200kg) and Unit Count (2000 units). We excluded these columns to streamline processing.
2. **Strict Zone Isolation:** A picker cannot switch zones within a single picklist. If an order spans multiple zones, it is effectively treated as separate sub-orders during the picking phase.
3. **Fragile Item Segregation:** Fragile items require special handling and cannot be mixed with heavy, normal items to prevent damage. They are processed in exclusive picklists with a reduced weight capacity (50kg).
4. **Operational Buffer:** While shifts are hourly, we assume a "Human Error & Delay Buffer" of 10 minutes. Therefore, a standard 1-hour slot is treated as **50 minutes of effective picking and staging time**.
5. **Setup Cost:** Every picklist incurs a fixed "Setup and Teardown" cost of **4 minutes** (2 min travel to zone + 2 min travel to staging). Therefore, fewer, longer picklists are mathematically superior to many short picklists.

Key Assumptions:

Component	Rationale / Strategic Goal	Impact on Throughput
Long picklists outperform many small picklists	Core "Long Picklist Strategy": Minimizes non-productive time (travel to zone/staging).	Saves 4–6 min of repeated walking and loading per batch, converting dead time into picking time.
50-min safety cap	Operational Buffer: Allocates 10 minutes per hour for unforeseen delays and human error.	Protects against real-world disruptions (jammed racks, human fatigue), ensuring theoretical schedules hold in practice.
Zone Priority Queue	Prioritization: Processes zones based on total SKU demand (quantity_counts).	High-volume zones are always cleared first, preventing bottlenecks and accelerating overall throughput.
SKU-first packing	Sorting Bias: Orders are sorted in descending order of order_qty within each zone.	Reduces bin hopping and unnecessary movement, improving walking efficiency and packing density.
Time-aware splitting	Resource Management: Picklist duration is dynamically set based on available picker counts per time window (e.g., 140 min for P1/P2, 50 min for P4/P5).	Seamlessly adapts when picker strength drops in later shifts, maximizing resource utilization at all times.
Fragile isolation	Constraint Enforcement: Separates fragile items into exclusive, reduced-capacity (50kg) picklists.	Prevents costly breakage and mis-picks, upholding inventory safety and reducing negative value from damaged goods.
Proactive Load Balancing ("Look-Ahead")	Idle Capacity Utilization: Assigns orders from *future* priority pods when the current pod queue is empty.	Smooths out demand spikes, prevents picker idle time, and reduces pressure on later, potentially lower-availability shifts.

Algorithm & Strategy

Core Philosophy: "The Long Picklist Strategy"

To maximize throughput, we minimize the frequency of non-productive travel (Zone <-> Staging). By consolidating tasks into the longest possible picklists (up to 140 minutes), we save approximately 4 minutes of setup/teardown time per cycle compared to shorter assignments.

Algorithmic Logic

Step 1: Pre-processing & Hierarchical Grouping

- **Data Cleaning:** We first optimize the dataset by removing physical dimension columns irrelevant to the primary constraints: Length, Width, and Height.
- **Grouping Strategy:** The data is grouped hierarchically:
    1. **Primary Grouping:** By pod_priority (P1 > P2 > ... > P9) to respect cutoff urgency.
    2. **Secondary Grouping:** By zone to strictly enforce the constraint that a picklist must belong to only one zone.
- **Sorting:** Within these groups, refined data is sorted in **descending order of order_qty**. This ensures that larger, high-volume orders are processed first, reducing the complexity of the remaining pool.

Step 2: Dynamic Time-Window Assignment

We treat picker shifts as dynamic resource windows. The maximum duration of a picklist is determined by the density of the workforce available in that specific pod_priority window:

- **High Availability (P1 - P3):** ~80 pickers available (Night Shift 1 & 2). We assign **140-minute** picklists to maximize utilization.

- **Transition Period (P4):** Workforce drops from 80 to 35. We reduce the limit to **50 minutes** to manage the shift handover.
- **Day Shifts (P5 - P9):** Standard **50-minute** limits to maintain a steady flow.

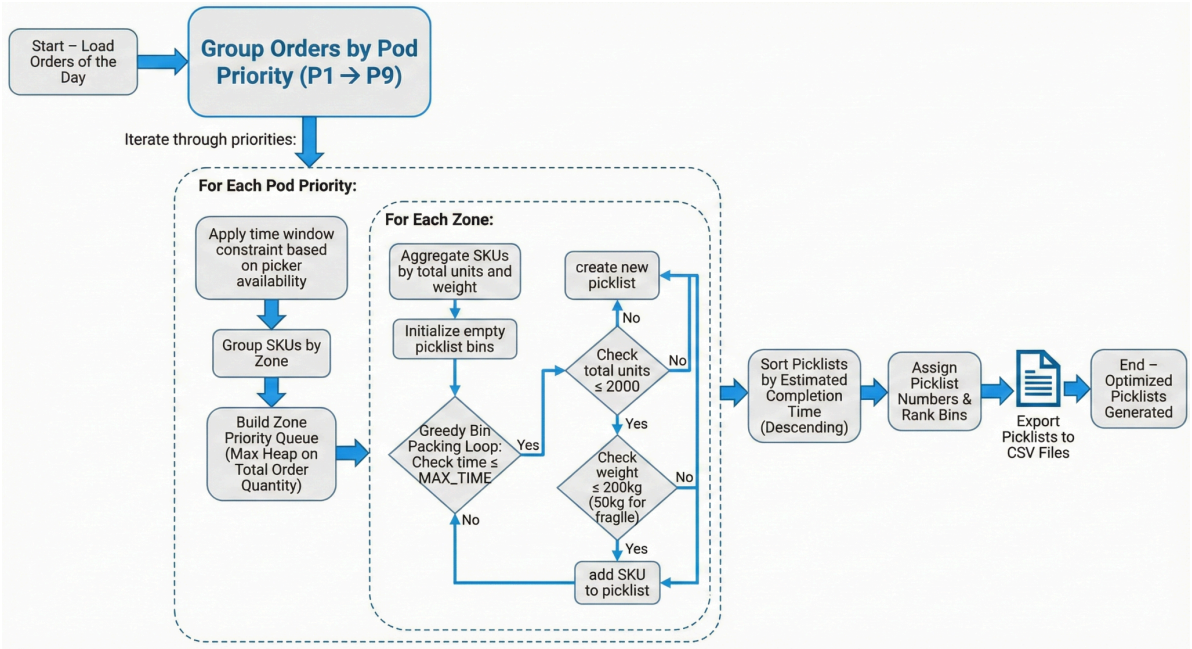
Priority	Shift Coverage	Available Pickers	Max Picklist Duration Strategy
P1	Night Shift 1 & 2 (9 PM - 11:30 PM)	~80	**140 mins** (Maximize duration to clear backlog)
P2	Night Shift 1 & 2 (11:30 PM - 2 AM)	~80	**140 mins** (High throughput window)
P3	Night Shift 1 & 2 (2 AM - 4 AM)	~80	**110 mins** (Full utilization of overlap)
P4	Mixed (4 AM - 6 AM)	80 -> 35	**50 mins** (Reduced duration to manage shift drop-off)
P5	Night Shift 2 (6 AM - 7 AM)	35	**50 mins** (Short cycles for final shift hour)
P6	Morning Shift (8 AM - 9 AM)	40	**50 mins** (New shift ramp-up)
P9	Morning + General (10 AM - 11 AM)	70	**50 mins** (High capacity, standard cycles)

Step 3: Proactive Load Balancing (Look-Ahead) A critical feature of our algorithm is **Idle Capacity Utilization**.

- We identified that during high-availability windows (e.g., P1/P2), pickers often cleared the current queue and remained idle.
- **Optimization:** Instead of leaving pickers idle, we will assign picklists from *next* priority groups (e.g., picking P2 orders during the P1 window) effectively "banking" work and reducing pressure on future time slots where picker counts might be lower.

Step 4: Constraint Satisfaction We construct picklists using a greedy approach that respects:

- **Unit Cap:** Max 2000 units.
- **Weight Cap:** Max 200kg for Normal, 50kg for Fragile.
- **Shift Boundaries:** Picklists are split if they would force a picker to work past their shift end time



PickList Class – Attributes and Functions Explanation

Class Attributes

POD_PRIORITIES

Defines the priority order in which store orders (pods) are processed.
Pods are processed from highest urgency (P1) to lowest urgency (P9).
This ensures time-critical orders are always handled first.

PICKER_SHIFTS

Stores all picker shift details including:
• Shift start and end time
• Number of pickers working in that shift
This structure is used to dynamically determine how many pickers are available at any given time.

Instance Attributes

Self.date: The specific date for which picklists are generated.

Self.dataframe: Contains all order records for the given day.

Self.pod_dict: Dictionary mapping each pod priority to its respective filtered dataframe. This enables independent processing for each cutoff window.

Core Functions

init(self, dataframe, date): Initializes the PickList object by loading the orders of the day and grouping them pod-wise into self.pod_dict.

build_zone_dict(self, pod_priority): Groups the pod-specific dataframe into warehouse zones and sorts SKUs inside each zone by descending demand. This becomes the base structure for picklist generation.

display_data(self, pod_priority, n=5): Displays the first *n* rows of the selected pod dataframe for debugging and inspection.

zone_counts(self, pod_priority): Returns the number of orders per zone to identify the most active warehouse zones.

quantity_counts(self, pod_priority): Returns total SKU units per zone. Used to prioritize zones with the highest workload.

build_zone_priority_queue(self, pod_priority): Builds a max-heap priority queue of zones ordered by total SKU quantity. This ensures zones with maximum picking demand are processed first.

print_zone_priority_queue(self): Prints the zone priority queue in descending order of workload.

Generate_picklist(self, pod_priority): This is the core algorithm. For each zone it:

- Aggregates SKU quantities
- Packs SKUs into picklists using greedy bin-packing
- Enforces all constraints:
  - Time limit per pod
  - Max 2000 units
  - Max 200kg weight (50kg for fragile)
- Generates optimized picklists per zone.

calculate_picklist_time(self, picklist_df): Estimates total picklist execution time using:

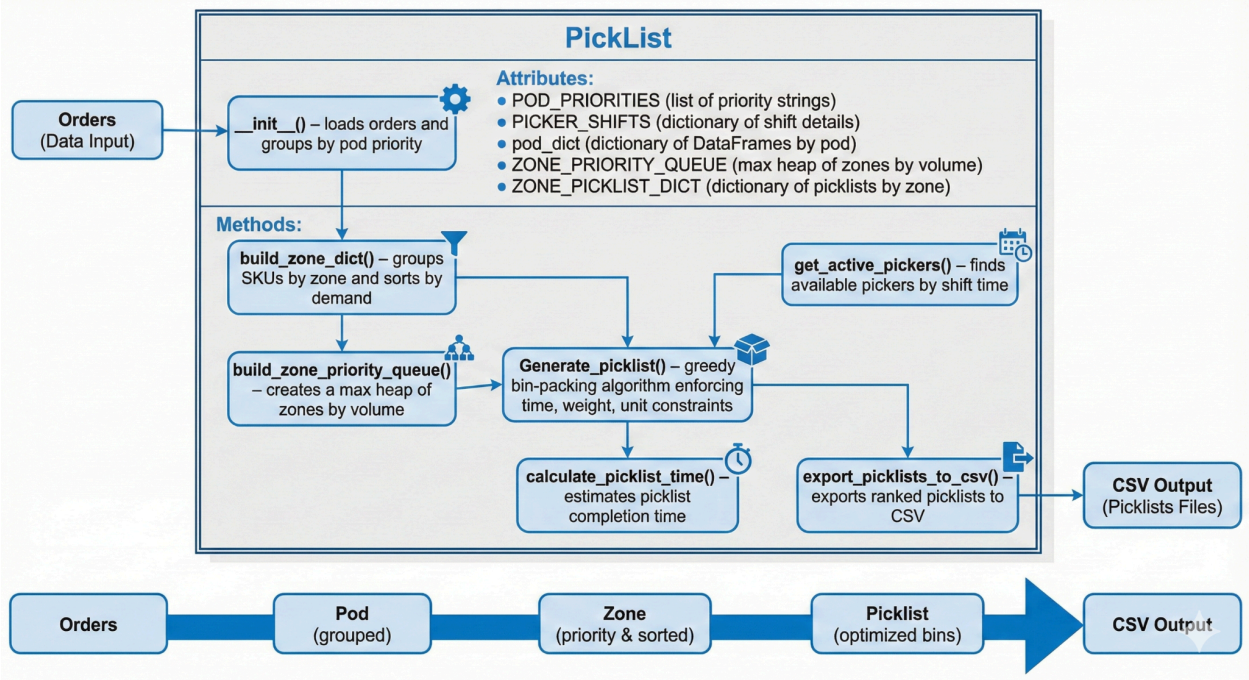
- Fixed travel times
- SKU-level movement time
- Unit pickup time

• Order unloading time.
Ensures picklists complete before cutoff.

display_generate_picklist_output(self): Displays all generated picklists along with:

• Total units
• Total weight
• Estimated completion time.

export_picklists_to_csv(self, output_dir): Exports picklists in the required hackathon format
{date}_{rank}_{picklist_no}.csv with proper bin ranking and metadata.



Key Insights & Trade-offs

Key Insights

1. **Workforce Re-Engineering Required:** Running our algorithm on the month-long dataset revealed a structural inefficiency: the current shift timings do not align perfectly with peak demand. We identified that a **new shift system** is required to reallocate pickers to heavy-demand timezones, specifically ramping up earlier to meet the P4 (6:00 AM) and P5 (7:00 AM) cutoffs where failure risks are highest.
2. **The "Look-Ahead" Efficiency:** By allowing pickers to process future pod_priority orders when the current queue is empty, we smoothed out demand spikes. This prevented the "crunch time" scenarios where pickers would otherwise be overwhelmed, significantly improving the Total units successfully picked metric.
3. **Minimizing "Dead Walking" Time:** The "Long Picklist" strategy saves ~24 minutes of travel time over a 2.5-hour block compared to short cycles. This directly translates to picking capacity for hundreds of additional units per picker per shift.
4. **Robustness via the "50-Minute Hour":** Limiting standard picklists to 50 minutes (instead of 60) creates a necessary buffer for warehouse realities (human error, blockage), ensuring that "theoretical" schedules hold up in "practical" execution.

Trade-offs

- **Throughput vs. Flexibility:**
    - *Trade-off:* Long picklists (140 mins) lock pickers into a zone, making it harder to react to sudden high-priority changes.
    - *Justification:* Since cutoffs are known in advance, maximizing volume is more valuable than flexibility.
- **Fragile Splitting:**
    - *Trade-off:* Segregating Fragile items increases the total number of picklists and travel instances.
    - *Justification:* This is an unavoidable operational cost to ensure inventory safety; mixing heavy and fragile items would result in damaged goods (negative value).
- **Sorting Bias:**
    - *Trade-off:* prioritizing large orders (descending order_qty) might delay the completion of many small, single-item orders.
    - *Justification:* Large orders carry more "weight" in terms of volume. Clearing them early simplifies the physical logistics of the warehouse floor.

Real World Application of our Algorithm: The Dynamic Dispatcher

Concept Overview

While our PickList class acts as the **Architect**—designing the most efficient batches of work—the assignment simulation acts as the **Field Commander**.

In a real-world warehouse, a static schedule (e.g., "John does List A at 9:00, List B at 9:50") is fragile. If John is fast, he sits idle. If he is slow, the whole schedule delays. Our solution replaces this with a **Dynamic, Event-Driven Dispatch System**.

How It Works in Practice

Imagine a digital "Dispatcher" standing at the staging area. It holds two critical pieces of information:

1. **The Master Queue:** A prioritized line of all generated picklists (sorted by Zone Urgency and Volume).
2. **The Live Roster:** A real-time tracker of every picker's status (Working or Available).

1. The "Hot-Swap" Assignment (Zero Idle Time)

Our algorithm utilizes a **Min-Heap Priority Queue** to track worker availability.

- **The Simulation:** The moment a worker finishes a task (e.g., at 9:45 PM), the system *immediately* identifies them as "Free."
- **Real World Impact:** As a picker drops off a completed cart at the staging area, their handheld device instantly buzzes with the next optimal picklist. There is zero "dead time" waiting for a supervisor to assign work.

2. Respecting Human Constraints (Shift Logic)

Efficiency cannot come at the cost of operational feasibility. The algorithm is "Shift-Aware."

- **The Logic:** Before assigning a 50-minute task, the system checks: *"Does this worker have at least 50 minutes left in their shift?"*
- **Real World Impact:** This prevents **"Zombie Tasks"**—orders that are started but abandoned halfway because the picker's shift ended. If a worker has only 10 minutes left, the system will not assign a long task, ensuring that every started picklist is a completed picklist.

3. Sequential Zone Clearing

We do not assign tasks randomly. The system pulls from the ZONE_PICKLIST_QUEUE, which we pre-sorted by workload volume.

- **The Logic:** The algorithm feeds the "heaviest" zones to the workforce first.
- **Real World Impact:** This ensures that the most congested areas of the warehouse are attacked by the fresh workforce at the start of the shift (9 PM), preventing bottlenecks from piling up closer to the cutoff times.

The Business Value

By automating this assignment process, we achieve:

- **Maximized Throughput:** Workers are utilized for 100% of their available operational minutes.
- **Scalability:** The logic holds true whether you have 10 pickers or 1,000.
- **Predictability:** Management gets an immediate "Finish Time" forecast. We know exactly when the backlog will be cleared based on current staff levels.