**Senga SDE System Architecture: Consolidation-Optimized Sequential Decision Engine**

**System Context Diagram**

**External Actors and Systems**

**1. Shippers (e.g., Tropical Heat)**

* **Interactions IN:**
  + Submit orders via API (shipper → multiple retail destinations)
  + Provide shipment specifications (weight, volume, fragility, time windows)
  + Specify delivery priorities and SLA requirements
  + Receive consolidation assignments and pickup schedules
* **Interactions OUT:**
  + Get pickup time windows
  + Receive shipment tracking updates (manual waypoint-based)
  + Get delivery confirmation with proof of delivery
  + Access billing and invoice data

**2. Retail Customers (e.g., Naivas, Chandarana, Carrefour)**

* **Interactions IN:**
  + Receive consolidated deliveries (multiple shippers in one truck)
  + Provide receiving time windows and dock availability
  + Confirm deliveries and report issues
* **Interactions OUT:**
  + Pre-delivery notifications (consolidated manifest)
  + Delivery ETA updates
  + Post-delivery feedback

**3. Drivers (Hired/Contracted Trucks)**

* **Interactions IN:**
  + Receive consolidated route manifests with:
    - Pickup sequence from multiple shippers
    - Delivery sequence to multiple retailers
    - Load consolidation instructions (which items go where)
    - Optimized routing considering mesh network
  + Manual GPS updates at waypoints
* **Interactions OUT:**
  + Report pickup completions (per shipper)
  + Report delivery completions (per retail customer)
  + Update truck capacity status
  + Flag issues (delays, capacity overruns, access problems)

**4. Fleet Manager**

* **Interactions IN:**
  + Monitor truck utilization (target: >75%)
  + View consolidation efficiency metrics
  + Receive capacity planning recommendations
* **Interactions OUT:**
  + Update truck availability and specs
  + Approve/modify strategic fleet allocation
  + Set operational constraints

**5. Operations Team**

* **Interactions IN:**
  + Monitor real-time consolidation decisions
  + Review exception alerts (low utilization, missed consolidations)
  + Access consolidation analytics and what-if scenarios
* **Interactions OUT:**
  + Override consolidation decisions when necessary
  + Adjust consolidation parameters (min utilization thresholds)
  + Validate learning system recommendations

**External Systems**

**6. Order Management System (Senga OMS)**

* **Interactions:**
  + **IN:** Real-time order stream via API
  + **IN:** Order updates, cancellations, modifications
  + **OUT:** Consolidation assignments back to OMS
  + **OUT:** Optimized pickup/delivery schedules
  + **Format:** REST API with webhook callbacks

**7. Shipper APIs (Multiple)**

* **Interactions:**
  + **IN:** Order submissions, shipment details
  + **OUT:** Pickup scheduling, consolidation notifications
  + **OUT:** Tracking updates (waypoint-based)

**8. Google Places API**

* **Interactions:**
  + **IN:** Address validation and geocoding
  + **IN:** Route distance/duration estimates
  + **OUT:** Validated addresses with GPS coordinates
  + **Constraint:** Rate limits, API costs

**9. Notification Service**

* **Interactions:**
  + **OUT:** SMS/Email to shippers and retailers
  + **OUT:** Driver app notifications

**Senga SDE System Boundary**

**Core Problem Statement**

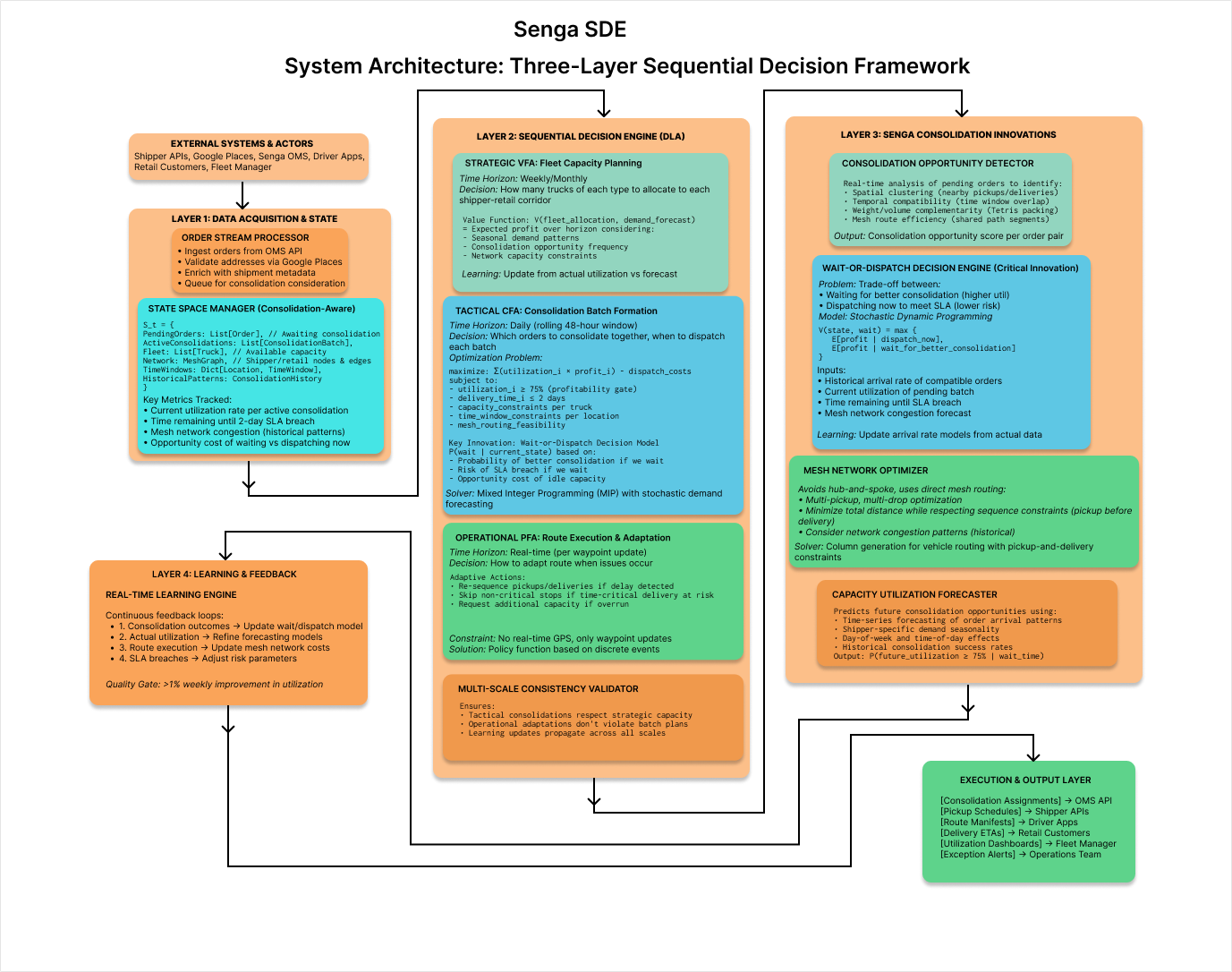
**Sequential Decision Problem:** Given a stream of small orders from multiple shippers to multiple retail destinations, dynamically decide:

1. **Which orders to consolidate** into each truck run
2. **When to dispatch** (vs. waiting for more consolidation opportunities)
3. **Which route to take** through the mesh network
4. **How to sequence pickups and deliveries** to maximize efficiency

Subject to:

* **Profitability constraint:** Capacity utilization ≥75%
* **Time constraint:** All deliveries within 2 days of order receipt
* **No warehousing:** Direct mesh routing only
* **Capacity constraint:** Truck weight/volume limits
* **Time window constraints:** Shipper pickup hours, retailer receiving hours

System Architecture: Three-Layer Sequential Decision Framework



**Key Architectural Decisions**

1. State Space Design (Consolidation-Centric)

@dataclass

class ConsolidationState:

    # Pending orders awaiting consolidation

    pending\_orders: List[Order]  # Each with origin, destination, weight, volume, time\_window

    # Active consolidation batches (not yet dispatched)

    active\_batches: List[ConsolidationBatch]  # Each tracking current utilization

    # Fleet availability

    available\_trucks: List[Truck]  # With capacity specs

    en\_route\_trucks: List[Truck]  # With current waypoint

    # Network state

    mesh\_graph: NetworkGraph  # Shipper/retail nodes, historical edge costs

    # Time constraints

    current\_time: datetime

    sla\_deadlines: Dict[OrderID, datetime]  # 2-day deadline per order

    # Learning state

    historical\_patterns: ConsolidationHistory  # Past success rates, arrival patterns

    utilization\_forecast: UtilizationModel  # Predicted future opportunities

2. Action Space Design

@dataclass

class ConsolidationAction:

    # Primary decision

    decision\_type: Literal["WAIT", "DISPATCH", "REJECT"]

    # If DISPATCH:

    batch\_assignment: Dict[OrderID, TruckID]

    pickup\_sequence: List[ShipperID]

    delivery\_sequence: List[RetailID]

    mesh\_route: List[Waypoint]

    # If WAIT:

    wait\_duration: timedelta

    target\_utilization: float  # Desired util before dispatch

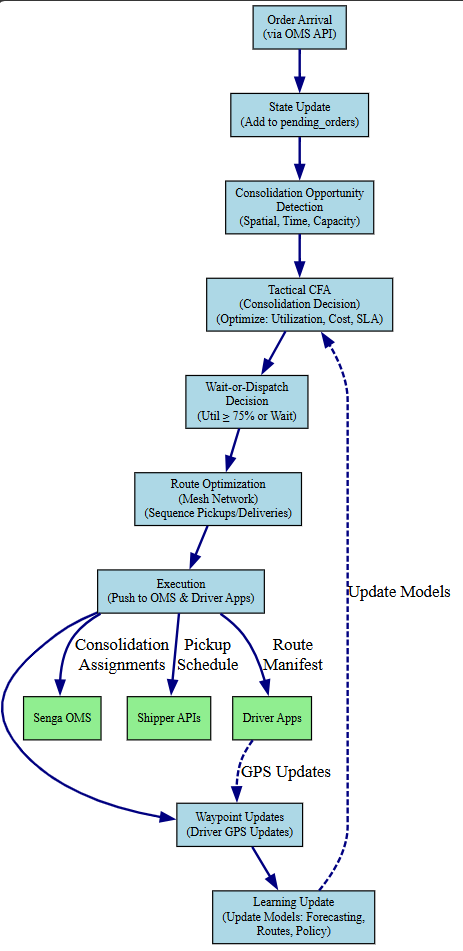
    # Metadata

    expected\_utilization: float

    expected\_profit: float

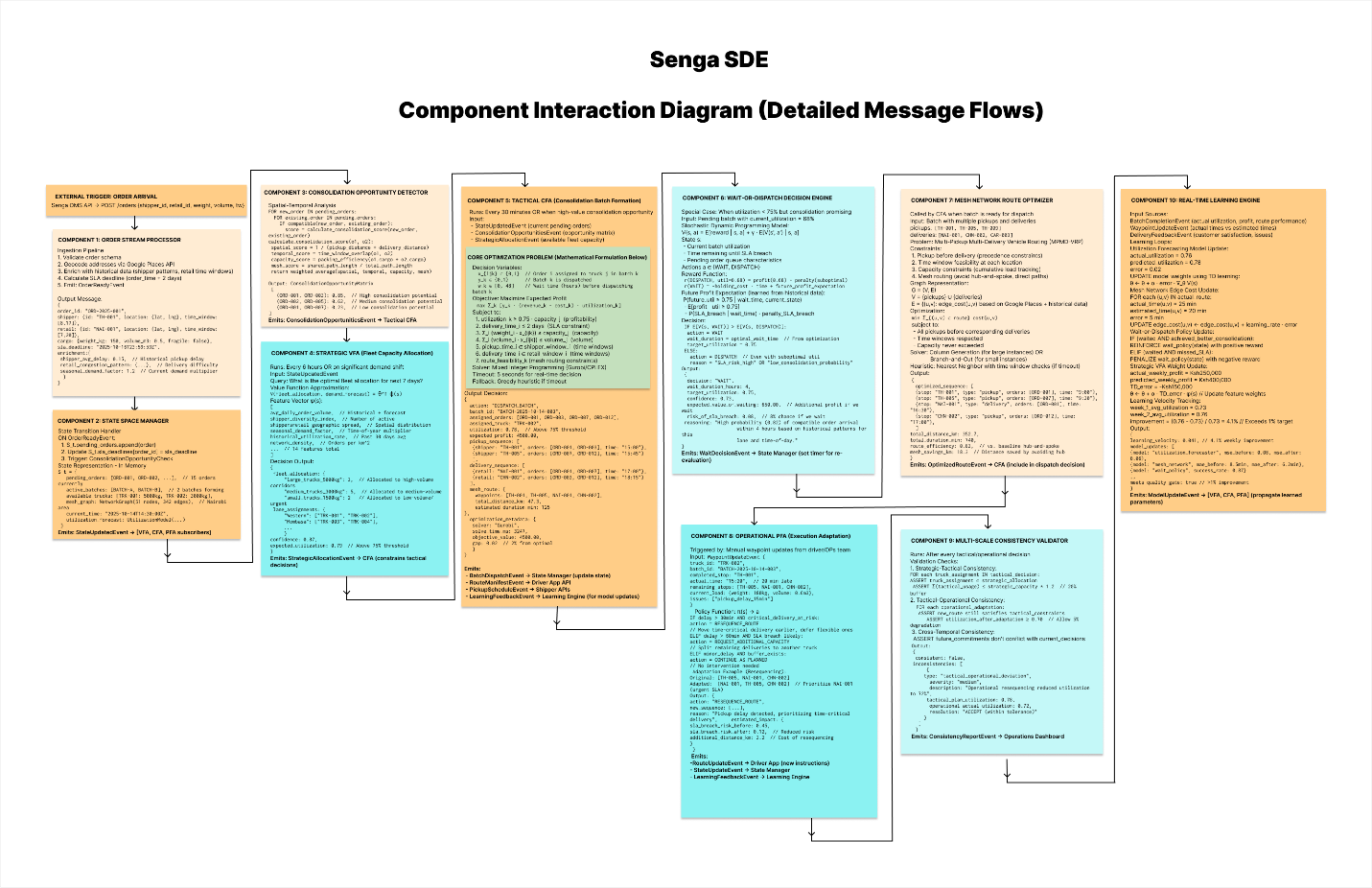
    risk\_of\_sla\_breach: float

3. Sequential Decision Flow (Critical Path)



Senga SDE Component Interaction Diagram & Mathematical Formulation

Part 1: Component Interaction Diagram (Detailed Message Flows)

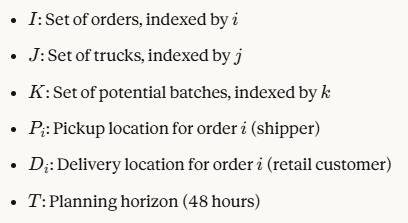


**Part 2: Mathematical Formulation of Core Consolidation Problem**

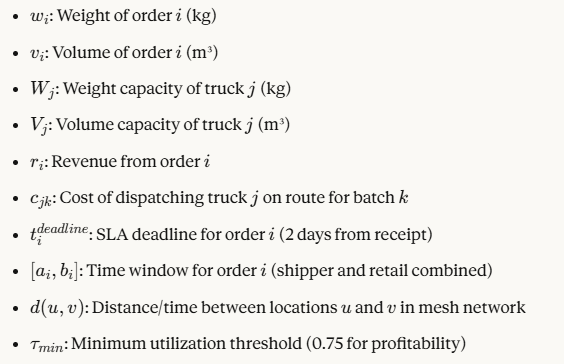
**Problem Statement**

Given a set of orders arriving over time, decide which orders to consolidate into truck runs such that profit is maximized while respecting capacity, time window, and utilization constraints.

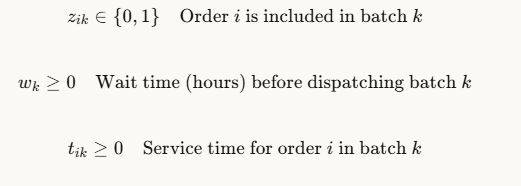
**Sets and Indices**

****

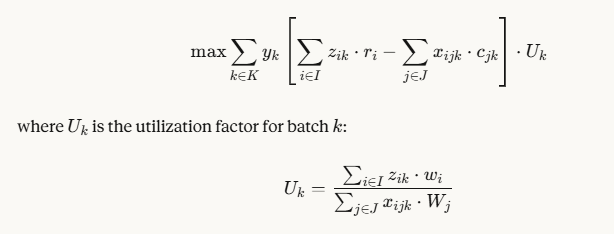
**Parameters**



Decision Variables

****

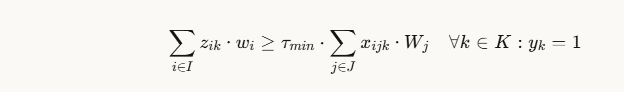
**Objective Function**

****

**Interpretation:** Maximize total profit across all batches, weighted by utilization (incentivizes high-utilization batches).

**Constraints**

**1. Profitability Constraint (Critical for Senga)**

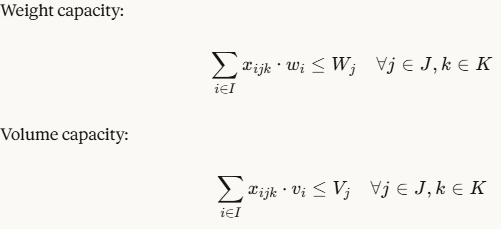
****

**Meaning:** If batch *k* is dispatched, weight utilization must be ≥75%.

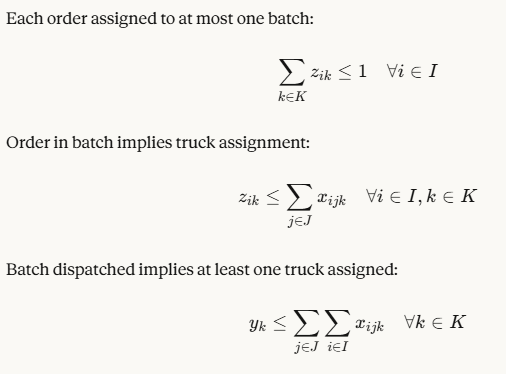
Volume variant:

****

**2. Capacity Constraints**



**3. Order Assignment Constraints**



**4. SLA (Two-Day Delivery) Constraints**

****

**Meaning:** Order service time plus wait time cannot exceed 2-day deadline.

**5. Time Window Constraints**

Pickup time window:

****

Delivery time window:

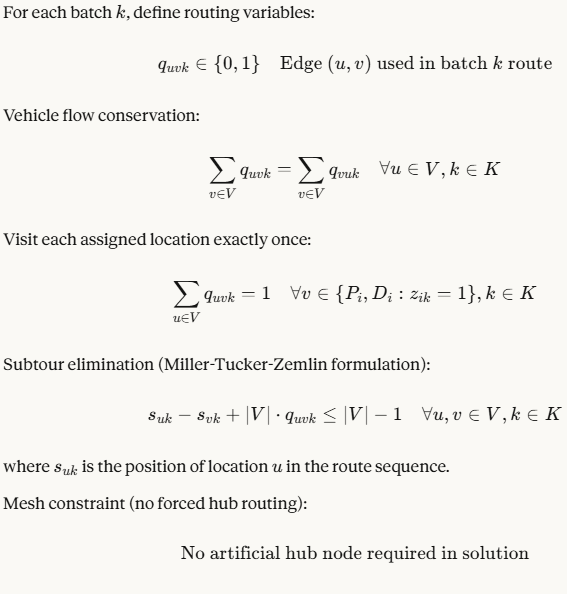


Precedence (pickup before delivery):

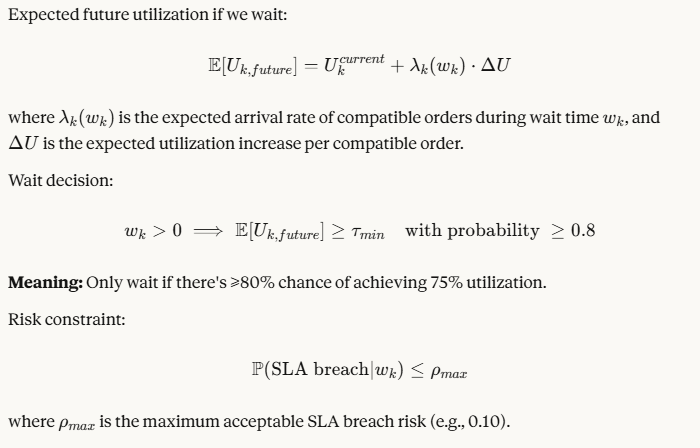


where *τPi*​​ is the service time at pickup location.

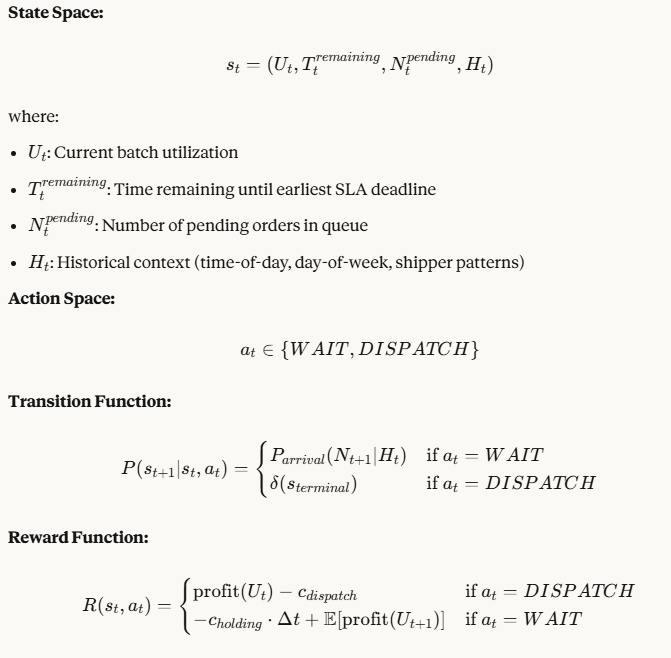
**6. Mesh Routing Constraints**

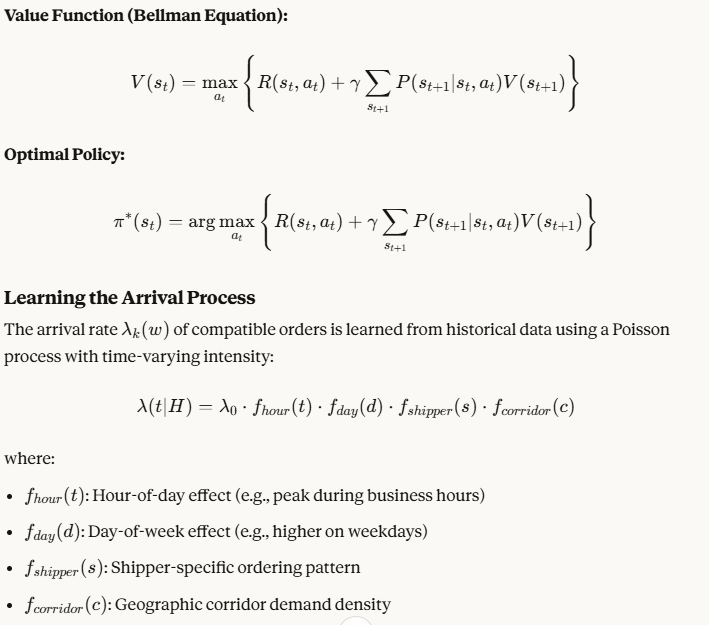


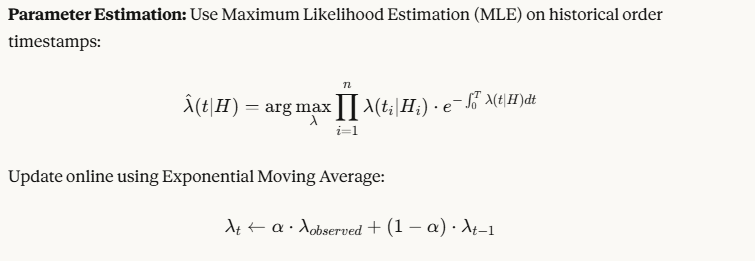
**7. Wait-or-Dispatch Decision Constraints**



**Stochastic Elements (Wait-or-Dispatch Model)**

**Markov Decision Process F**

****



**Solution Approach**

**Two-Stage Stochastic Programming**

**Stage 1 (Tactical CFA):** Decide on batch composition and dispatch timing **Stage 2 (Operational PFA):** Adapt to realized demand and execution issues

**Formulation:**



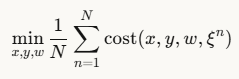
subject to Stage 1 constraints (above), where *ξ* represents random variables (future order arrivals, travel times, delays).

**Sample Average Approximation (SAA):**

Generate *N* scenarios of future demand:



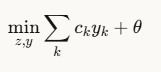
Solve deterministic equivalent:

****

**Decomposition for Scalability**

**Benders Decomposition:**

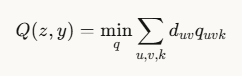
Master Problem (batch assignment):



subject to:

* Order assignment constraints
* *θ≥Q(z,y) (*subproblem optimal value)

Subproblem (routing for fixed batch):



subject to:

* Mesh routing constraints for batch *k*
* Time window constraints

Iterate until convergence.

ALGORITHM: ConsolidationBatchOptimizer

INPUT:

  - pending\_orders: Set of unfulfilled orders

  - available\_trucks: Set of trucks with capacity specs

  - current\_time: Timestamp

  - historical\_data: Past consolidation patterns

OUTPUT:

  - batch\_decisions: List of (orders, truck, route, dispatch\_time)

PROCEDURE:

1. OPPORTUNITY\_DETECTION:

   FOR each pair (o\_i, o\_j) in pending\_orders:

     score[o\_i, o\_j] = consolidation\_score(o\_i, o\_j)

   clusters = spatial\_temporal\_clustering(pending\_orders, score)

2. FOR each cluster in clusters:

   2.1 BATCH\_FORMATION:

       Initialize batch\_k with cluster orders

       current\_util = calculate\_utilization(batch\_k)

   2.2 WAIT\_OR\_DISPATCH\_DECISION:

       IF current\_util >= 0.75:

         decision = DISPATCH

       ELSE:

         // Stochastic optimization

         V\_wait = expected\_value\_of\_waiting(batch\_k, historical\_data)

         V\_dispatch = profit(current\_util) - dispatch\_cost

         IF V\_wait > V\_dispatch AND sla\_risk\_acceptable:

           decision = WAIT

           wait\_time = optimal\_wait\_duration(batch\_k)

           schedule\_reevaluation(batch\_k, current\_time + wait\_time)

         ELSE:

           decision = DISPATCH

   2.3 IF decision == DISPATCH:

       2.3.1 TRUCK\_ASSIGNMENT:

             truck = assign\_optimal\_truck(batch\_k, available\_trucks)

       2.3.2 ROUTE\_OPTIMIZATION:

             pickups = [order.pickup\_location for order in batch\_k]

             deliveries = [order.delivery\_location for order in batch\_k]

             route = solve\_MPMD\_VRP(

               pickups, deliveries, truck.capacity,

               mesh\_network\_graph, time\_windows

             )

       2.3.3 VALIDATION:

             IF NOT validate\_sla\_compliance(route, batch\_k):

               REJECT batch\_k

               Continue

             IF NOT validate\_time\_windows(route):

               route = adaptive\_resequencing(route)

             utilization\_final = calculate\_utilization(batch\_k)

             IF utilization\_final < 0.75:

               LOG warning: "Suboptimal batch dispatched"

       2.3.4 OUTPUT:

             batch\_decisions.append({

               orders: batch\_k,

               truck: truck,

               route: route,

               dispatch\_time: current\_time,

               utilization: utilization\_final

             })

3. CONSISTENCY\_CHECK:

   FOR each batch\_decision in batch\_decisions:

     validate\_multi\_scale\_consistency(batch\_decision)

4. LEARNING\_UPDATE:

   FOR each batch\_decision in batch\_decisions:

     queue\_for\_learning\_feedback(batch\_decision)

RETURN batch\_decisions

END PROCEDURE

```

---

## Complexity Analysis

### Computational Complexity

\*\*Consolidation Opportunity Detection:\*\* $O(|I|^2)$

- Pairwise comparison of all pending orders

\*\*Batch Formation (MIP):\*\* $O(2^{|I|} \cdot |J| \cdot |K|)$ (worst case)

- NP-hard in general, but practically solvable for small instances with modern solvers

- Typical instance: 50 orders, 10 trucks, 20 batches → solvable in seconds

\*\*Mesh Route Optimization (MPMD-VRP):\*\* $O(|V|!)$ (worst case)

- NP-hard, but branch-and-cut or heuristics yield good solutions quickly

- Typical instance: 6 pickups, 8 deliveries → solvable in milliseconds

\*\*Wait-or-Dispatch MDP:\*\* $O(|S| \cdot |A| \cdot T)$ per evaluation

- Discrete state space makes this tractable

- Value iteration converges in $O(T \cdot |S|^2)$

\*\*Total Decision Time Budget:\*\* 5 seconds for real-time operation

- If MIP timeout, fall back to greedy heuristic

- Heuristic guarantees feasible solution in $O(|I| \cdot \log |I|)$

