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

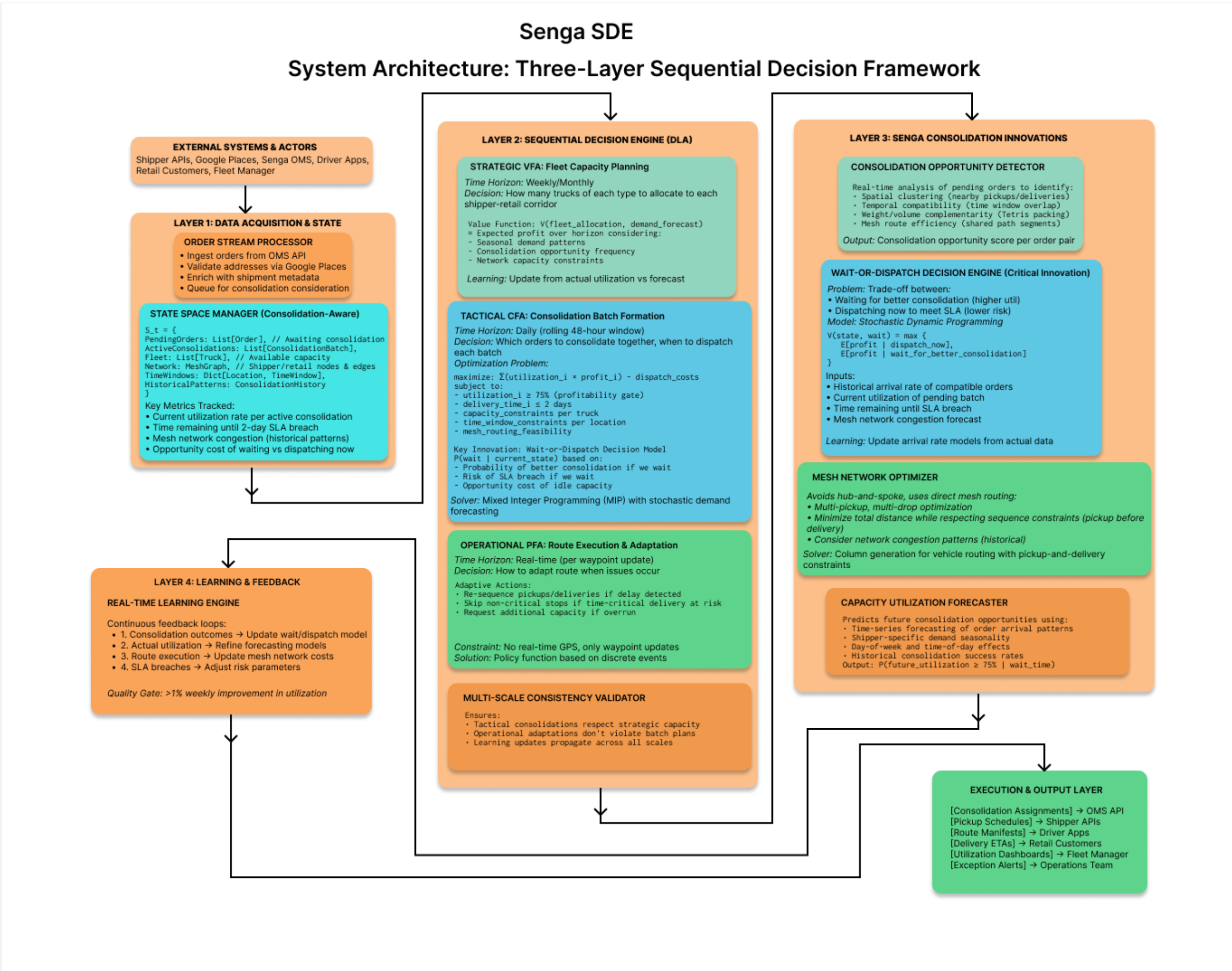
Sequential Decision Problem: Given a stream of small orders from multiple shippers to multiple retail destinations, dynamically decide:

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

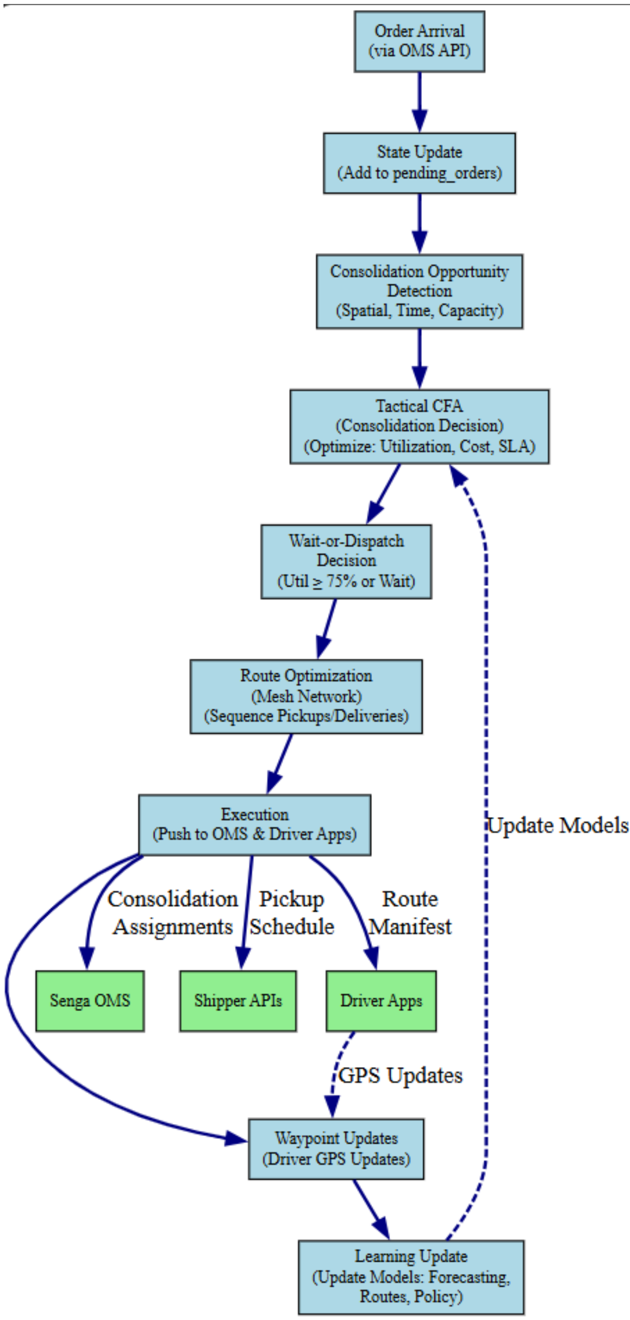
```
@dataclass
class ConsolidationAction:
    # Primary decision
    decision_type: Literal["WAIT", "DISPATCH", "REJECT"]

    # 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
```

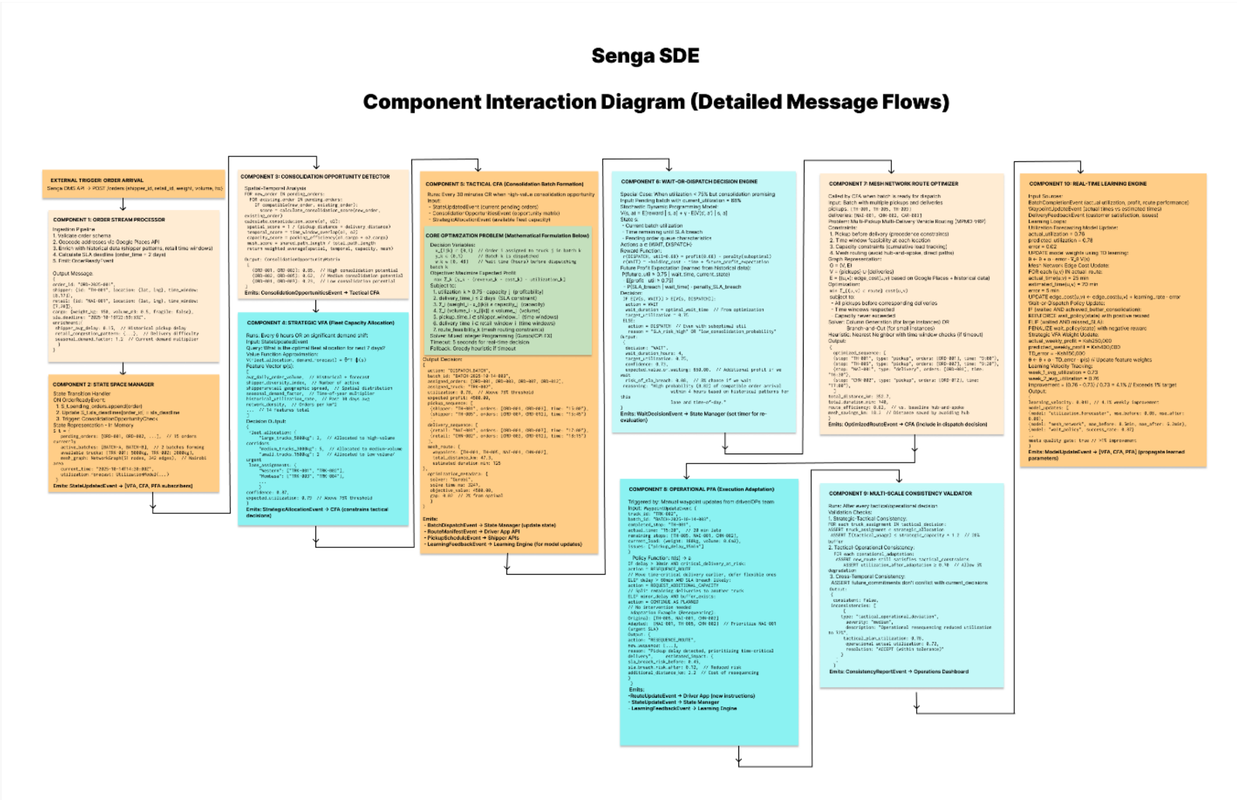
```
# Metadata
expected_utilization: float
expected_profit: float
risk_of_sla_breach: float
```

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

- I: Set of orders, indexed by i
- J: Set of trucks, indexed by j
- K: Set of potential batches, indexed by k
- P_i: Pickup location for order i (shipper)
- D_i: Delivery location for order i (retail customer)
- T: Planning horizon (48 hours)

Parameters

- w_i: Weight of order i (kg)
- v_i: Volume of order i (m³)
- W_j: Weight capacity of truck j (kg)
- V_j: Volume capacity of truck j (m³)
- r_i: Revenue from order i
- c_{jk}: Cost of dispatching truck j on route for batch k
- $t_i^{deadline}$: SLA deadline for order i (2 days from receipt)
- $[a_i, b_i]$: Time window for order i (shipper and retail combined)
- $d(u, v)$: Distance/time between locations u and v in mesh network
- τ_{min}: Minimum utilization threshold (0.75 for profitability)

Decision Variables

$$z_{ik} \in \{0,1\} \quad \text{Order } i \text{ is included in batch } k$$

$$w_k \geq 0 \quad \text{Wait time (hours) before dispatching batch } k$$

$$t_{ik} \geq 0 \quad \text{Service time for order } i \text{ in batch } k$$

Objective Function

$$\max \sum_{k \in K} y_k \left[\sum_{i \in I} z_{ik} \cdot r_i - \sum_{j \in J} x_{ijk} \cdot c_{jk} \right] \cdot U_k$$

where U_k is the utilization factor for batch k:

$$U_k = \frac{\sum_{i \in I} z_{ik} \cdot w_i}{\sum_{j \in J} x_{ijk} \cdot W_j}$$

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

Constraints

1. Profitability Constraint (Critical for Senga)

$$\sum_{i \in I} z_{ik} \cdot w_i \geq \tau_{min} \cdot \sum_{j \in J} x_{ijk} \cdot W_j \quad \forall k \in K : y_k = 1$$

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

Volume variant:

$$\sum_{i \in I} z_{ik} \cdot v_i \geq \tau_{min} \cdot \sum_{j \in J} x_{ijk} \cdot V_j \quad \forall k \in K : y_k = 1$$

2. Capacity Constraints

Weight capacity:

$$\sum_{i \in I} x_{ijk} \cdot w_i \leq W_j \quad \forall j \in J, k \in K$$

Volume capacity:

$$\sum_{i \in I} x_{ijk} \cdot v_i \leq V_j \quad \forall j \in J, k \in K$$

3. Order Assignment Constraints

Each order assigned to at most one batch:

$$\sum_{k \in K} z_{ik} \leq 1 \quad \forall i \in I$$

Order in batch implies truck assignment:

$$z_{ik} \leq \sum_{j \in J} x_{ijk} \quad \forall i \in I, k \in K$$

Batch dispatched implies at least one truck assigned:

$$y_k \leq \sum_{j \in J} \sum_{i \in I} x_{ijk} \quad \forall k \in K$$

4. SLA (Two-Day Delivery) Constraints

$$t_{ik} + w_k \leq t_i^{deadline} \quad \forall i \in I, k \in K : z_{ik} = 1$$

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

5. Time Window Constraints

Pickup time window:

$$a_{P_i} \leq t_{P_i,k} \leq b_{P_i} \quad \forall i \in I, k \in K : z_{ik} = 1$$

Delivery time window:

$$a_{D_i} \leq t_{D_i,k} \leq b_{D_i} \quad \forall i \in I, k \in K : z_{ik} = 1$$

Precedence (pickup before delivery):

$$t_{P_i,k} + \tau_{P_i} + d(P_i, D_i) \leq t_{D_i,k} \quad \forall i \in I, k \in K : z_{ik} = 1$$

where τ_{Pi} is the service time at pickup location.

6. Mesh Routing Constraints

For each batch k, define routing variables:

$$q_{uvk} \in \{0,1\} \quad \text{Edge } (u,v) \text{ used in batch } k \text{ route}$$

Vehicle flow conservation:

$$\sum_{v\in V} q_{uvk} = \sum_{v\in V} q_{vuk} \quad \forall u \in V, k \in K$$

Visit each assigned location exactly once:

$$\sum_{u\in V} q_{uvk} = 1 \quad \forall v \in \{P_i, D_i : z_{ik} = 1\}, k \in K$$

Subtour elimination (Miller-Tucker-Zemlin formulation):

$$s_{uk} - s_{vk} + |V| \cdot q_{uvk} \leq |V| - 1 \quad \forall u, v \in V, k \in K$$

where s_{uk} is the position of location u in the route sequence.

Mesh constraint (no forced hub routing):

$$\text{No artificial hub node required in solution}$$

7. Wait-or-Dispatch Decision Constraints

Expected future utilization if we wait:

$$\mathbb{E}[U_{k,future}] = U_k^{current} + \lambda_k(w_k) \cdot \Delta U$$

where $\lambda_k(w_k)$ is the expected arrival rate of compatible orders during wait time w_k, and ΔU is the expected utilization increase per compatible order.

Wait decision:

$$w_k > 0 \implies \mathbb{E}[U_{k,future}] \geq \tau_{min} \quad \text{with probability } \geq 0.8$$

Meaning: Only wait if there's ⩾80% chance of achieving 75% utilization.

Risk constraint:

$$\mathbb{P}(\text{SLA breach} | w_k) \leq \rho_{max}$$

where ρ_{max} is the maximum acceptable SLA breach risk (e.g., 0.10).

Stochastic Elements (Wait-or-Dispatch Model)

Markov Decision Process F

State Space:

$$s_t = (U_t, T_t^{remaining}, N_t^{pending}, H_t)$$

where:

- U_t: Current batch utilization
- $T_t^{remaining}$: Time remaining until earliest SLA deadline
- $N_t^{pending}$: Number of pending orders in queue
- H_t: Historical context (time-of-day, day-of-week, shipper patterns)

Action Space:

$$a_t \in \{WAIT, DISPATCH\}$$

Transition Function:

$$P(s_{t+1}|s_t, a_t) = \begin{cases} P_{arrival}(N_{t+1}|H_t) & \text{if } a_t = WAIT \\ \delta(s_{terminal}) & \text{if } a_t = DISPATCH \end{cases}$$

Reward Function:

$$R(s_t, a_t) = \begin{cases} \text{profit}(U_t) - c_{dispatch} & \text{if } a_t = DISPATCH \\ -c_{holding} \cdot \Delta t + \mathbb{E}[\text{profit}(U_{t+1})] & \text{if } a_t = WAIT \end{cases}$$

Value Function (Bellman Equation):

$$V(s_t) = \max_{a_t} \left\{ R(s_t, a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) V(s_{t+1}) \right\}$$

Optimal Policy:

$$\pi^*(s_t) = \arg\max_{a_t} \left\{ R(s_t, a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) V(s_{t+1}) \right\}$$

Learning the Arrival Process

The arrival rate $\lambda_k(w)$ of compatible orders is learned from historical data using a Poisson process with time-varying intensity:

$$\lambda(t|H) = \lambda_0 \cdot f_{hour}(t) \cdot f_{day}(d) \cdot f_{shipper}(s) \cdot f_{corridor}(c)$$

where:

- $f_{hour}(t)$: Hour-of-day effect (e.g., peak during business hours)
- $f_{day}(d)$: Day-of-week effect (e.g., higher on weekdays)
- $f_{shipper}(s)$: Shipper-specific ordering pattern
- $f_{corridor}(c)$: Geographic corridor demand density

Parameter Estimation: Use Maximum Likelihood Estimation (MLE) on historical order timestamps:

$$\hat{\lambda}(t|H) = \arg\max_{\lambda} \prod_{i=1}^{n} \lambda(t_i|H_i) \cdot e^{-\int_0^T \lambda(t|H)dt}$$

Update online using Exponential Moving Average:

$$\lambda_t \leftarrow \alpha \cdot \lambda_{observed} + (1 - \alpha) \cdot \lambda_{t-1}$$

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:

$$\min_{x,y,w} \mathbb{E}_{\xi}[\text{cost}(x, y, w, \xi)]$$

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:

$$\xi^1, \xi^2, \ldots, \xi^N \sim P_{historical}$$

Solve deterministic equivalent:

$$\min_{x,y,w} \frac{1}{N} \sum_{n=1}^{N} \text{cost}(x, y, w, \xi^n)$$

Decomposition for Scalability

Benders Decomposition:

Master Problem (batch assignment):

$$\min_{z,y} \sum_{k} c_k y_k + \theta$$

subject to:

- Order assignment constraints
- $\theta \geq Q(z,y)$ (subproblem optimal value)

Subproblem (routing for fixed batch):

$$Q(z, y) = \min_{q} \sum_{u,v,k} d_{uv} q_{uvk}$$

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 $O(|I| \cdot \log |I|)$
````
````

Senga SDE

Data Flow Diagram (Level 1 - Major Processes)