Write-Up: Multi-Echelon Supply Chain Environment

1 Multi-Echelon Supply Chain Environment

1.1 1. States (S_t)

At any time t, the environment's state is composed of four primary components, plus the time index:

1. **On-hand inventory** for each node i.
   Denoted by $I_{i,t}$, for $i \in \{\text{market}, \text{retailers}, \text{distributors}, \text{producers}\}$.

2. **Pipeline inventory** for each route $(i \to j)$.
   Denoted as $T_{(i,j),t}$. This represents materials currently in transit between nodes i and j. These are tracked for each time lag up to the route's lead time.

3. **Sales and backlog** for each retailer-to-market route $(r \to 0)$.
   Sales $S_{(r,0),t}$ measure how many units were actually sold at time t. *Backlog* $B_{(r,0),t}$ measures outstanding (unfilled) demand carried over from previous steps.

4. **Demand** for each retailer route $(r \to 0)$ at time t.
   Denoted by $D_{(r,0),t}$. This may be drawn from a random process, for example a normal or Poisson distribution.

5. **Time index** t.
   The horizon runs from $t = 0$ to $t = T$.

Formally, the environment's state can be written as:

$$S_t = \big(I_t,\, T_t,\, S_t,\, B_t,\, D_t,\, t\big).$$

1.2 2. Other Known Information

In addition to the above states, the environment has parameters that remain fixed or are known at each time step:

- **Lead times $\ell_{(i,j)}$.**
  Each route (i,j) has a nonnegative integer lead time indicating how many steps inventory stays in transit.

- **Capacities and constraints:**

  - *Inventory capacities* inv_capacity_i for each node i.
  - *Reordering route capacity* $\text{reordering\_route\_capacity}_{(i,j)}$. This limits how much can be shipped along a route at one time step.

- **Costs and prices**:

  - $\text{inventory\_holding\_cost}_i$ for on-hand inventory at each node.
  - $\text{material\_holding\_cost}_{(i,j)}$ for pipeline (in-transit) inventory.
  - operating_cost_i for production nodes.
  - $\text{unit\_price}_{(i,j)}$ for sales along route (i,j). In particular, final sales to the market $(r \to 0)$ often fetch higher prices.
  - $\text{unfulfilled\_utility\_penalty}_{(r,0)}$ for unmet demand. This encourages fulfilling demand rather than stockpiling inventory.

- **Demand distribution**:
  Each retailer route $(r \to 0)$ has an associated demand process, for example:

  $$D_{(r,0),t} \;\sim\; \mathcal{N}(\mu,\, \sigma^2),$$

  or any other chosen distribution. These demands can be generated each time step.

- **Miscellaneous penalty constants**:

  - D: Large cost factor for out-of-bounds violations (e.g. negative reorders).
  - P: Additional penalty factor for large constraint violations.

1.3 3. Actions (A_t)

At each time step t, the agent chooses how many units to reorder (or produce) along each valid route $(i \to j)$. Concretely:

- **Reorder action** $R_{(i,j),t}$. This is a nonnegative real number, subject to route capacity and lead time. In code, the action space is a `Box` with one entry per route.

Hence the full action at time t is:

$$A_t \;=\; \bigl(R_{(i_1,j_1),t},\; R_{(i_2,j_2),t}, \dots\bigr),$$

covering all valid reordering routes.

1.4 4. Action Correction Functions

The environment automatically applies checks that modify actions if they are invalid or outside bounds, and imposes corresponding penalties.

1. **Sanitize actions:**
   If the chosen reorder amount $|R_{(i,j),t}|$ is below a small threshold ϵ, it is set to 0.0 to avoid tiny floating-point shipments.

2. **Bounds checking:**

   (a) If $R_{(i,j),t} < 0$, the action is clipped to 0 and the environment adds a penalty $\big(P + D \cdot (\text{difference})^2\big)$.

   (b) If $R_{(i,j),t}$ exceeds the capacity $\text{reordering\_route\_capacity}_{(i,j)}$, it is clipped to that capacity and a similar penalty is applied.

3. **Observation bounding checks:**
   After the environment updates, if any state dimension is outside the allowed observation space (e.g. negative inventory or over capacity), the environment clips that dimension and applies a penalty $\big(P+D\cdot(\text{difference})^2\big)$.

These correction steps ensure the simulation does not break if the agent proposes unrealistic actions.

1.5 5. Transition Function

After an action A_t is chosen at time t, the environment transitions from S_t to S_{t+1} via the following updates:

1. **Record the reorder amounts**:

$$R_{(i,j),t} = (\text{sanitized and bounded action for each route } (i,j)).$$

   Shipments will arrive after the route's lead time $\ell_{(i,j)}$.

2. **Arrival of pipeline inventory**:
   For each route $(i \to j)$, the quantity that was shipped $\ell_{(i,j)}$ steps earlier arrives in node j. If $t' = t - \ell_{(i,j)} \geq 0$, then

$$R_{(i,j),t'} \;\to\; (\text{on-hand at } j \text{ at time } t+1).$$

3. **On-hand inventory update**:
   Each node's on-hand inventory is incremented by inbound arrivals and decremented by outbound shipments. For production nodes, we may include yields or operating costs here:

$$I_{j,t+1} \;=\; I_{j,t} + \big(\text{arrivals from all inputs to } j\big) \;-\; \big(\text{outflow or final sales}\big).$$

4. **Pipeline inventory update**:
   The environment tracks how much is in transit for each route:

$$T_{(i,j),t+1} \;=\; T_{(i,j),t} \;-\; \text{(arrivals)} \;+\; R_{(i,j),t}.$$

5. **Demand realization and satisfaction**:

   - Demand $D_{(r,0),t}$ is observed.
   - Retailers attempt to satisfy this demand from on-hand inventory.
   - Actual sales $S_{(r,0),t}$ is the minimum of on-hand and backlog+new demand.

$$S_{(r,0),t} = \min\Big(I_{r,t},\; B_{(r,0),t} + D_{(r,0),t}\Big).$$

   - Unsatisfied demand becomes backlog:

$$B_{(r,0),t+1} = B_{(r,0),t} + D_{(r,0),t} - S_{(r,0),t}.$$

6. **Advance the time index**:
   t increments to $t+1$. If $t \geq T$, the episode ends.

1.6 6. Reward and Cost Computation

At each time step t, the environment computes a "cost" and then returns a reward of Reward $= -$Cost. Several components go into this cost:

1. **Inventory holding cost**:

$$\text{cost}_{\text{inventory}} \;=\; \sum_{i} \Big(I_{i,t} \times \text{inventory\_holding\_cost}_i\Big).$$

2. **Operating cost** (for production nodes):

$$\text{cost}_{\text{operating}} \;=\; \sum_{\substack{\text{production node}\\ i}} \left(\frac{\text{outflow}}{\text{yield}_i}\right) \times \text{operating\_cost}_i.$$

3. **Pipeline holding cost**:

$$\text{cost}_{\text{pipeline}} \;=\; \sum_{(i,j)} \Big(T_{(i,j),t} \times \text{material\_holding\_cost}_{(i,j)}\Big).$$

4. **Backlog penalty**:

$$\text{cost}_{\text{backlog}} \;=\; \sum_{(r,0)} B_{(r,0),t+1} \times \text{unfulfilled\_utility\_penalty}_{(r,0)}.$$

5. **Revenue from sales**:

$$\text{cost}_{\text{sales}} \;=\; -\sum_{(i,j)} S_{(i,j),t} \times \text{unit\_price}_{(i,j)}.$$

(This contributes a negative cost, i.e. profit.)

6. **Violation penalties** from action bounding and observation bounding:

$$\text{cost}_{\text{violations}} \;=\; \sum \Big(P + D \cdot (\text{diff})^2 \Big).$$

Bringing them together:

$$\text{Cost}_t \;=\; \text{cost}_{\text{inventory}} + \text{cost}_{\text{operating}} + \text{cost}_{\text{pipeline}} + \text{cost}_{\text{backlog}} - \big(\text{sales revenue}\big) + \text{cost}_{\text{violations}}.$$

$$\text{Reward}_t \;=\; -\text{Cost}_t.$$

1.7 7. Episode Termination

1. **Horizon-based**: Once the time index t exceeds the final horizon T, the episode terminates.
2. **Other conditions** (optional): The environment may terminate if catastrophic states (e.g., large negative inventory) occur. In the provided code, termination is purely horizon-based.

2 Putting It All Together

The flow for each time step t is:

1. **Observe** state S_t.
2. **Select** action A_t (reorder amounts along each route).
3. **Sanitize & bound** actions to correct negative or oversized reorders.
4. **Update** pipeline, on-hand inventory, backlog, and demand.
5. **Calculate** cost and hence reward $\big(R = -\text{cost}\big)$.
6. **Clip** observations that are out of bounds (applying further penalties).
7. **Return** the next state S_{t+1} and reward R_t. If $t \geq T$, the episode ends.