Details on the RoRo Stevedoring Simulation

Cargo Availability Checks

At the beginning of stevedoring, all cargo units have processing status `initial` and predecessor status `blocked`. Each cargo unit then checks the processing status of its predecessors. If it has no predecessors, the predecessor status is set to `available`.

Whenever a tug selects, moves, finishes, or preempts a cargo, all unavailable cargo units check their predecessors' processing statuses and update their individual predecessor status:

- If at least one predecessor has processing status `initial`, the cargo's predecessor status remains `blocked`.
- When all predecessors have processing status `selected`, `in_process`, or `finished`, but at least one is in stage `selected`, a cargo unit updates its predecessor status to `pred_selected`. A cargo with predecessor status `pred_selected` can be `selected` by a tug that then travels to the cargo's departure position.
- When all predecessors have processing status `in_process` or `finished`, the cargo's predecessor status becomes `available`. With this status, it is guaranteed that the predecessors will be handled. The tug can start moving the cargo.

Tug Process Flow and Logic

The following describes a tug's default process:

- If the tug must finish handling an operation it began before the current simulation starts, it continues and completes the operation (new tug status: `idle`).
- If not in progress, the tug checks whether it has status `broken`. If not broken (i.e., tug status: `idle`), it checks its dedicated list of operations and gets the next planned cargo's predecessor status.

- If the cargo has predecessor status `available` or `pred_selected`, and handling the cargo would not issue a dead-end situation of blocked tugs, the tug selects this cargo and travels to the departure position (tug status: `moving_to_task`, cargo status: `selected`).
- After the travel time, it again checks the cargo's availability. If `available`, the tug proceeds.
- If the cargo unit is an import unit (i.e., tug and cargo are on the vessel), the tug checks if enough buffer time has passed to previous operations (e.g., to have enough space to actually get to the cargo), and waits as long as necessary (tug status: `delays_start`). This simplification neglects that waiting in reality would occur at a position close-by, not at the departure position itself.
- Next, the tug begins processing (tug status: `handling_cargo`; cargo status: `in_process`) by coupling the cargo and moving it to its destination position, where they then uncouple.
- When the processing time, including uncoupling, has passed, and the cargo is an export cargo (i.e., tug and cargo are on the vessel), the tug checks if all import predecessors have started and all export predecessors have finished. Further, it checks if enough buffer time has passed since these operations. If so, the cargo is moved successfully. Otherwise, the tug waits for as long as necessary (tug status: `delays_end`). This approach simplifies reality, where the uncoupling and waiting period would be reversed; from the simulation perspective, the order does not matter, as the total time until completion remains the same.
- The cargo's and tug's positions are updated to the cargo's destination position (tug status: `idle`; cargo status: `finished`).

Four logics prevent the tug process from failing:

- Breakdown logic: If a tug breaks after handling a cargo (tug status: `broken`), it signals the simulation controller its breakdown and ends its process.

- Skipping logic: (i) If the next planned cargo unit does not have predecessor status `available` or `pred_selected`, but (ii) it is not blocked by a broken tug, and (iii) the simulation parameters allow for changing the sequence, the tug randomly selects one of its assigned cargo units with the required status (with descending probability along the assigned sequence).
- Cannot continue logic: If a tug's next planned cargo is (i) blocked by a direct or indirect predecessor that is (ii) assigned to a broken tug, the tug calls replanning methods instead of selecting it.
- Preemption logic: If the tug traveled to a cargo unit, it must wait until the cargo's predecessor status becomes `available`. As long as this is not true, the tug checks if handling the cargo (indirectly) depends on a predecessor assigned to a broken tug after each update of cargo or tug statuses. If such a dependency exists, the tug signals the simulation controller that a tug broke down and the system crashed. Further, it preempts the cargo (tug status: `idle`; cargo statuses: `initial`/`blocked`) to provide the digital twin (DT) with more options to leave dead-end situations. The preemption further signals all succeeding cargo units to update their predecessor statuses to `blocked`.