Advanced Modeling for Operations

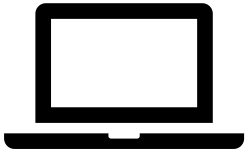
ASSIGNMENT – PART 1

POLITECNICO
MILANO 1863

DIPARTIMENTO DI
INGEGNERIA GESTIONALE

Course organisation
Assessment

The **exam** consists of**:**

- **Group assignment**  **70%**
- **Individual written test**  **30%**

- ***To pass the course***, *students should get a **positive evaluation** (i.e. ≥ 18) in **both the group assignment and the written test***
- *The **assignment grade** will be **valid for one academic year** (i.e., until September 2023)*

Assignment Timeline

September

October → **Assignment – Part 1**

Lessons dedicated to the assignment: October 17th and 24th

November

December → **Assignment – Part 2**

Lesson dedicated to the assignment: December 5th

Same operations setting (i.e., design of a factory logistics system with autonomous tugger trains)

January → **Deadline:** one week before the first call

Assignment
Problem description

Design of a factory logistics system with autonomous tugger trains

Warehousing and transportation of raw materials, components, and finished products within the factory

1. Storage and picking of raw materials and components
2. Transportation of raw materials and components to the production lines
3. Production of finished goods
4. Transportation of finished goods to the warehouse
5. Storage, picking and shipment of finished goods

Assignment
Problem description

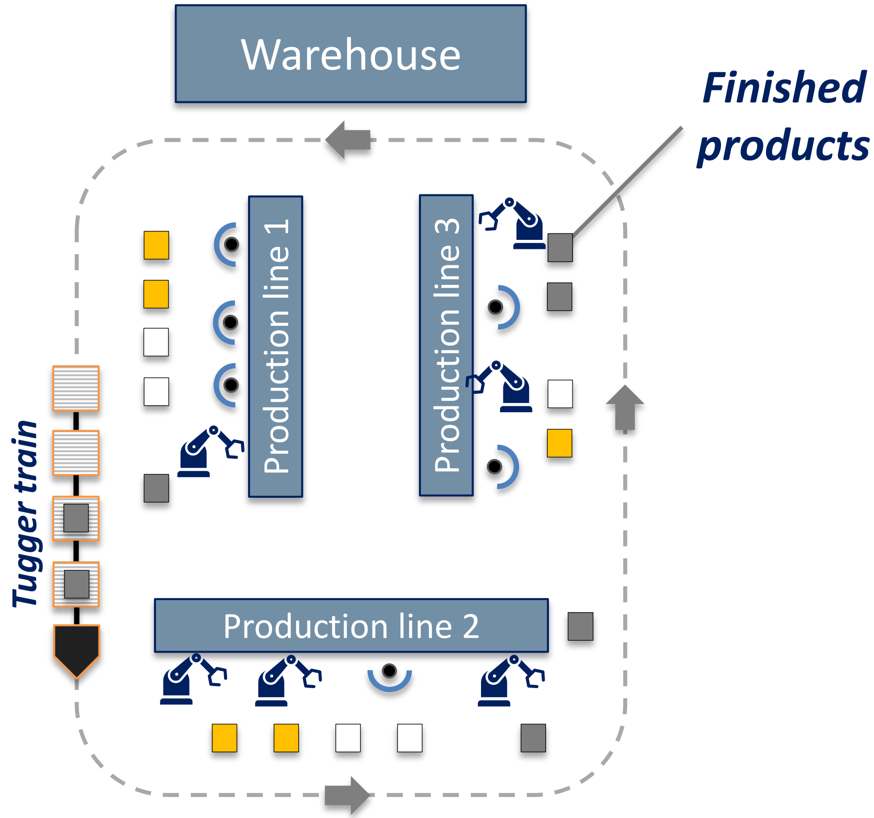
Design of a factory logistics system with autonomous tugger trains

Warehousing and transportation of raw materials, components, and finished products within the factory

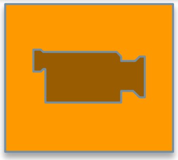
1. Storage and picking of raw materials and components
2. Transportation of raw materials and components to the production lines
3. **Production of finished goods**
4. **Transportation of finished goods to the warehouse**
5. Storage, picking and shipment of finished goods

Focus of the assignment

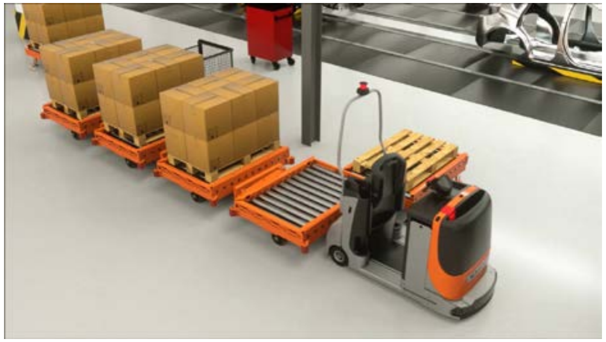
Design of a factory logistics system with autonomous tugger trains



Tugger train

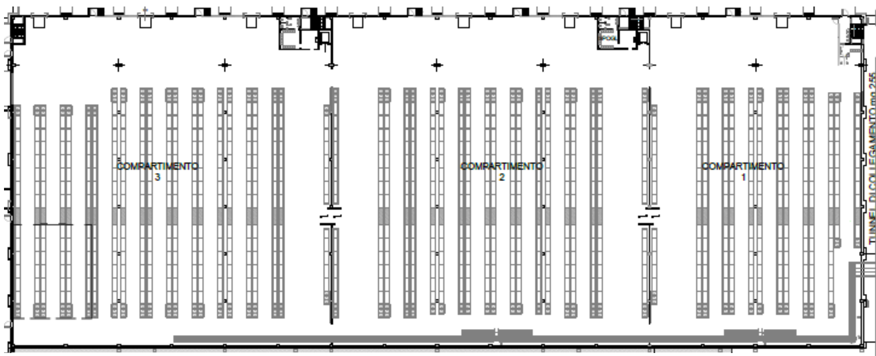


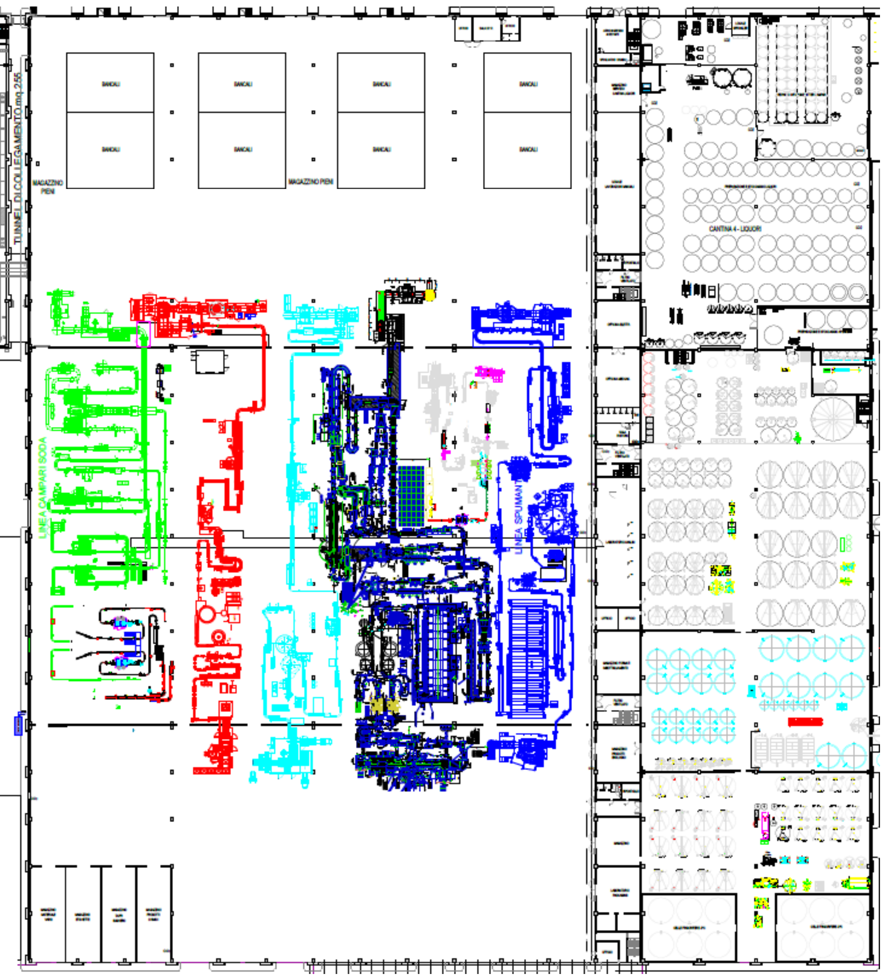
Design of a factory logistics system with autonomous tugger trains



- Each tugger train consists of **1 tugger vehicle and 4 wagons**
- The tugger vehicle is **automated** (**no driver** is needed)
- Tugger trains have a maximum **loading capacity**:
  - 4 unit loads
  - 2 tons (overall weight of the transported unit loads)
- Tugger vehicles are **electric vehicles**. Energy is stored in a **battery**, that needs to be periodically recharged at a **charging station**

Problem description: factory layout

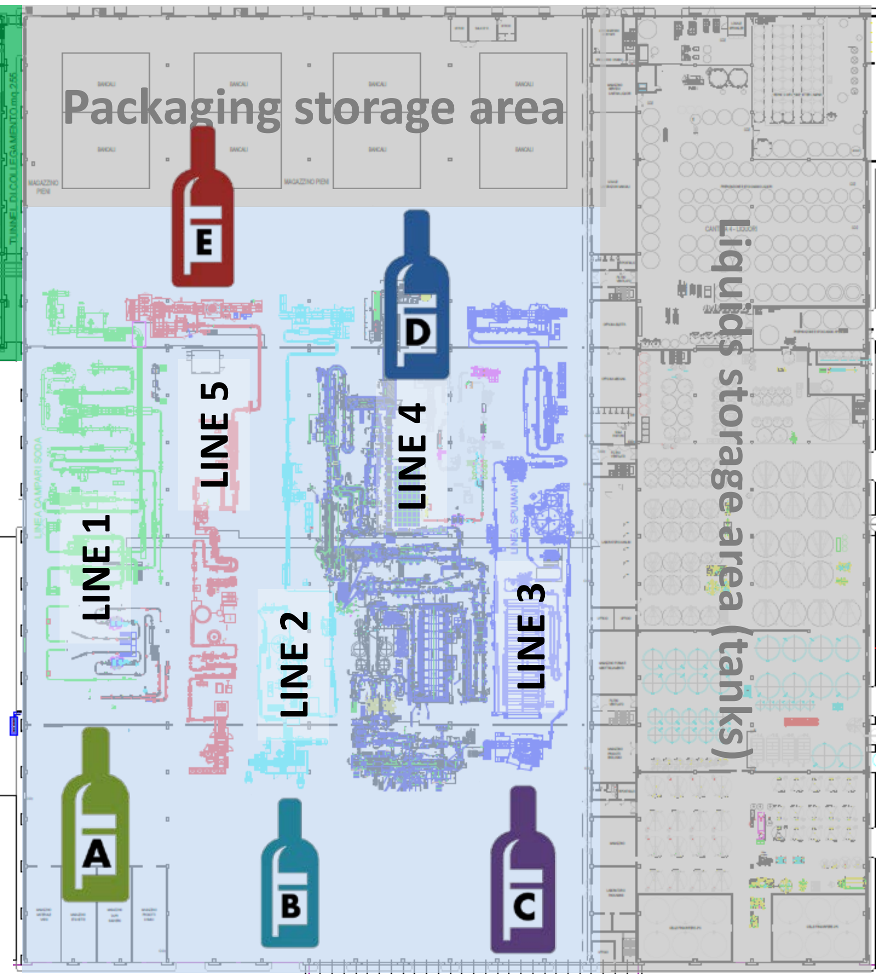




Bottling plant of a spirits manufacturer

working **two 8-hour shifts per day**

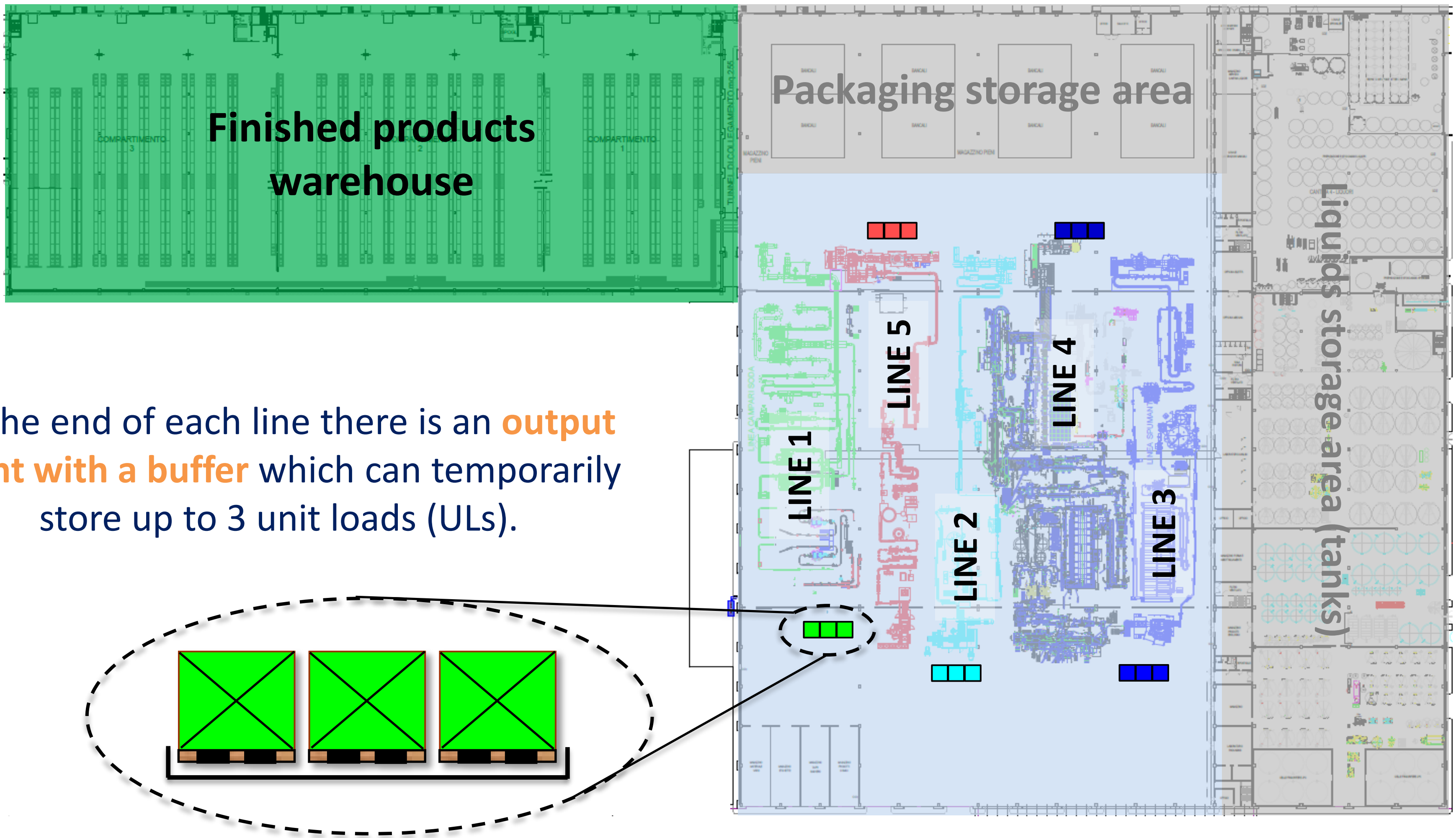
Problem description: factory layout



5 production lines (bottling lines)
making **different products**:

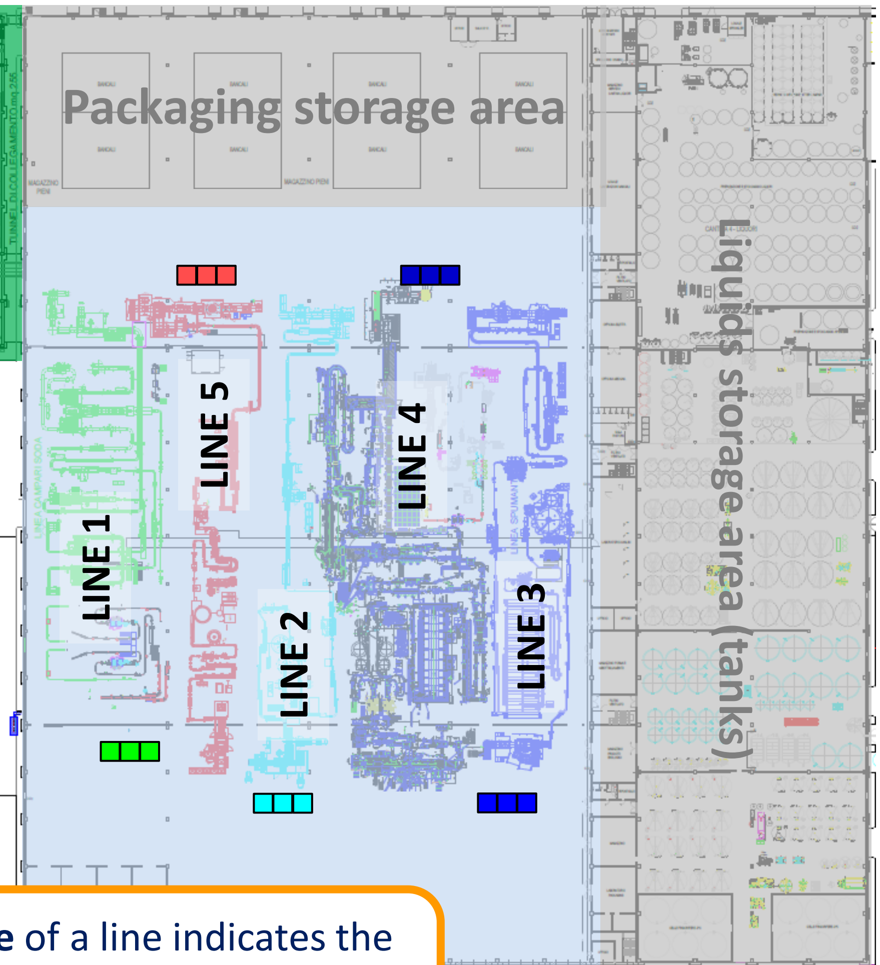
Line ID	Product ID	Production cycle time [minutes/unit load]	Weight [kg/unit load]
1	A	15	400
2	B	9	600
3	C	18	700
4	D	12	500
5	E	9	500

Assignment
Problem description: factory layout



At the end of each line there is an **output point with a buffer** which can temporarily store up to 3 unit loads (ULs).

Assignment
Problem description: factory layout

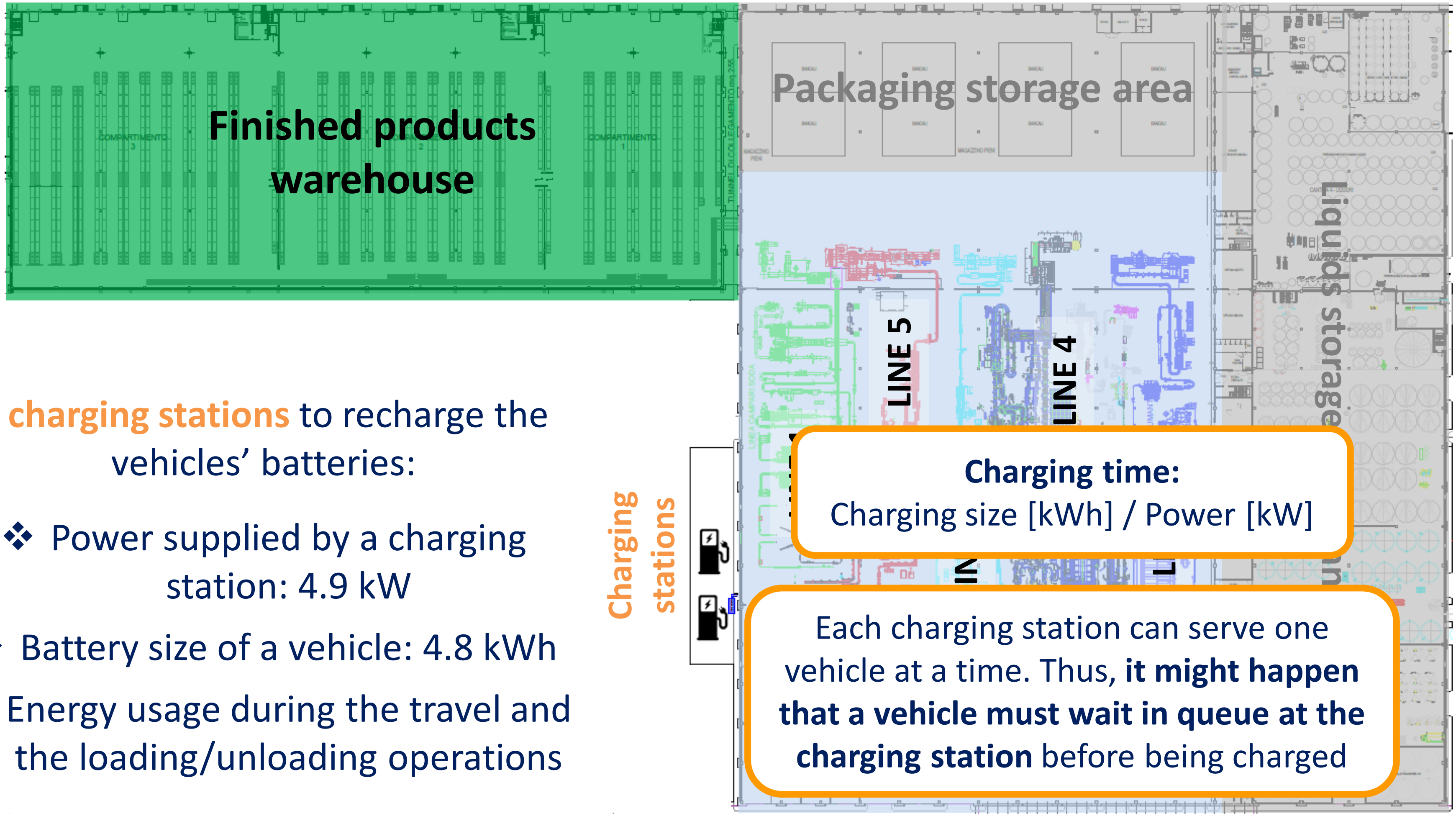


At the end of each line there is an **output point with a buffer** which can temporarily store up to 3 unit loads (ULs).

If the buffer is full, the line must stop. As soon as at least one UL is removed from the buffer, the line can resume its production activities.

The **idle time** of a line indicates the time during which the line is not producing (since the buffer is full)

Assignment
Problem description: factory layout



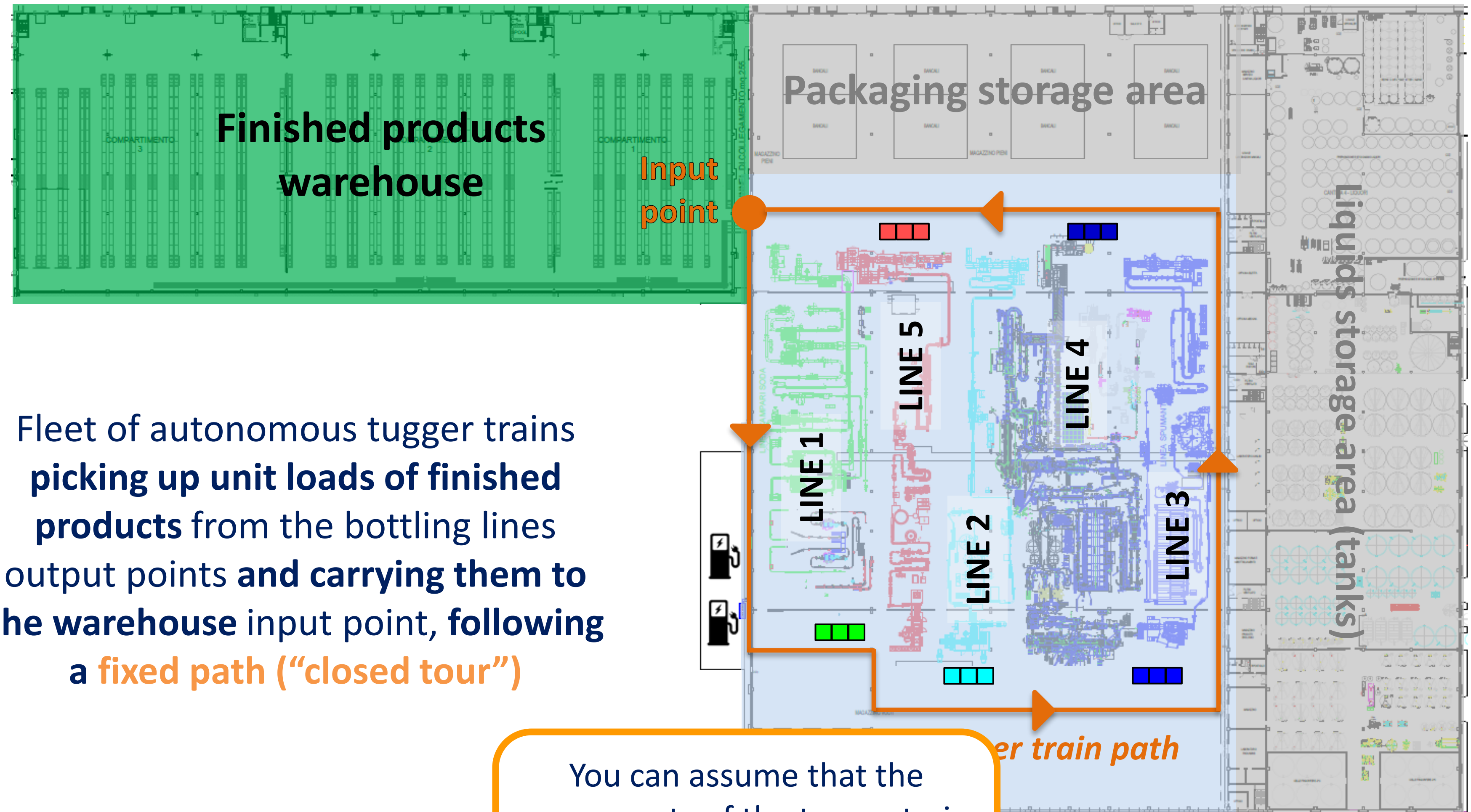
2 **charging stations** to recharge the vehicles' batteries:

- Power supplied by a charging station: 4.9 kW
- Battery size of a vehicle: 4.8 kWh
- Energy usage during the travel and the loading/unloading operations

Charging time:
Charging size [kWh] / Power [kW]

Each charging station can serve one vehicle at a time. Thus, **it might happen that a vehicle must wait in queue at the charging station** before being charged

Problem description: factory layout



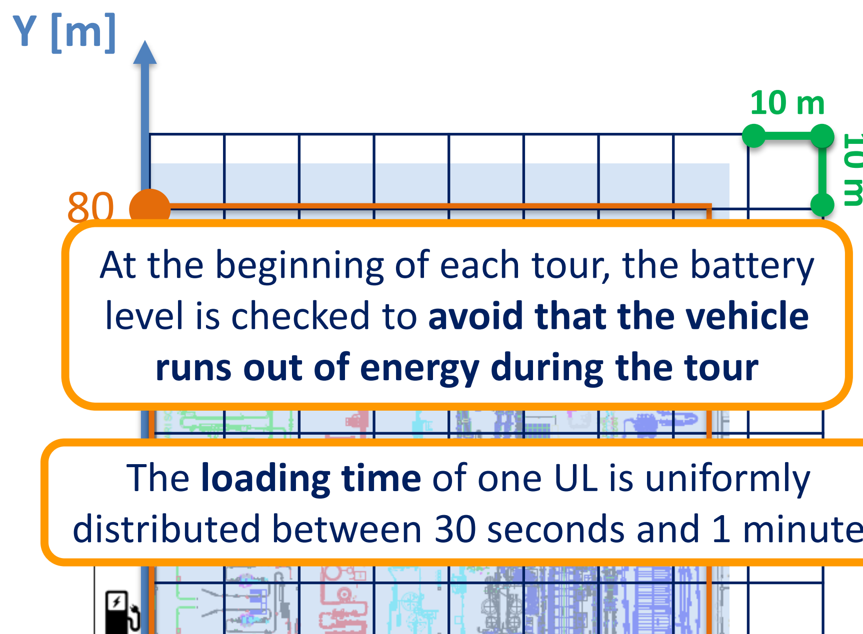
Fleet of autonomous tugger trains **picking up unit loads of finished products** from the bottling lines output points **and carrying them to the warehouse** input point, **following** a **fixed path ("closed tour")**

You can assume that the movements of the tugger train along the path are rectilinear
→ **Rectilinear distances**

Assignment
Problem description: factory layout

Each tugger train follows
a **fixed path (“closed tour”):**

1. It starts at the **warehouse input point**
2. (It goes to a **charging station**, if needed)
3. It passes by all the **lines output points**, checking if there are ULs to be picked up. If so, and if the train still has some capacity left, it picks up one or more ULs
4. It goes back to the **warehouse input point**, where it unloads the ULs



At the beginning of each tour, the battery level is checked to **avoid that the vehicle runs out of energy during the tour**

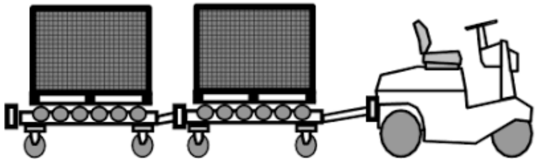
The **loading time** of one UL is uniformly distributed between 30 seconds and 1 minute

The **unloading time** of ULs at the warehouse input point has a fixed component (30 seconds to stop the vehicle in the right position) and a variable component (uniformly distributed between 30 s and 1 min per UL)

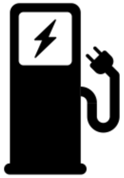
The coordinates of the lines output points, together with the cycle times, are reported in the file “lines_info.csv”

Assignment
Agents and methods

The agents are:



The tugger trains → **class Train**

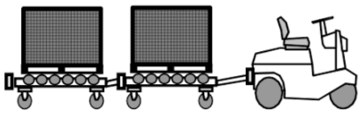


The charging stations → **class ChargingStation**

The bottling lines → **class Line**

Assignment
Agents: attributes and methods

Class Train



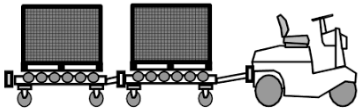
ATTRIBUTES

- Battery size [kWh] → ***battery_size***
- Current battery level [kWh] → ***remaining_energy***
- Maximum number of ULs that can be loaded on a tugger train → ***capacity***
- Number of ULs currently being carried → ***load***
- Coordinates of the current position → ***pos_x*** , ***pos_y***
- Coordinates of the next stop → ***next_stop_x*** , ***next_stop_y***
- Time at which the current task will be completed → ***task_endtime***
- Id of the next line output point to be visited → ***next_line***
- Boolean attribute indicating if the battery must be charged → ***need_to_charge***
- Station where the battery will be charged → ***selected_charging_station***

Assignment
Agents: attributes and methods

METHODS

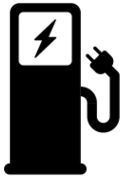
Class Train



Before starting a tour, the vehicle **decides whether going to a charging station** or proceeding to the first line output point	→ **check_charge()**
The vehicle **moves from the current position to the next stop**. If the next stop is a line, it also **loads the ULs**. If the next stop is the warehouse, it also **unloads the ULs**	→ **move()**
The vehicle **waits** for the charging station to be available **and then charges its battery**	→ **charging()**
If the previous task (movement, loading/unloading, charging) has ended and **if there is still time left** before the end of the simulation time, **the next task is launched**	→ **step()**

Assignment
Agents: attributes and methods

class ChargingStation



ATTRIBUTES

Waiting time before a vehicle can be charged → ***waiting_time***

METHODS

The charging station **checks if new vehicles have arrived and updates its waiting time** accordingly → **step()**

Assignment
Agents: attributes and methods

ATTRIBUTES

- ID number of the line → ***line_index***
- Cycle time (i.e., interval between the production of 2 ULs) → ***cycle_time***
- Maximum number of ULs in the buffer → ***buffer_size***
- Current number of ULs in the buffer → ***UL_in_buffer***
- Overall number of produced ULs → ***total_production***
- Overall time of production stoppage since the buffer was full → ***idle_time***
- Attribute needed to keep count of the cycle time → ***count_time***

class Line

METHODS

If the production of a UL has ended and the buffer is not full, **the production of the next UL starts** (**otherwise, the idle time is increased**) → **step()**

Assignment
Simulation model (“BASE CASE”)

Electric_vehicle_base_case_v4.py

```
from mesa import Agent, Model
from mesa.time import BaseScheduler #BaseScheduler activates all the agents at each step, one agen
import random

def read_line_info(): #Function that imports the coordinates of the lines output points from a fil
    x = []
    y = []
    cycle_times = [] #Time [minutes] to produce one unit load
    path = "lines_info.csv"
    h = open(path, "r")
    line_count = 0
    for line in h:
        if line_count == 0: #Ensures that the first line (containing col
            line_count += 1
            continue
        a = line.split(',') #The elements of the list a are strings, cor
        x.append(float(a[1]))
        y.append(float(a[2]))
        cycle_times.append(float(a[3]))
    h.close()
    return(x, y,cycle_times)

#Parameters:
warehouse_coord = {'x':0,'y':80} #Coordinates of the warehouse input poi
charging_stations_x = [0,0] #x coordinates of the first and second charg
charging_stations_y = [10,20] #y coordinates of the first and second cha
lines_output_points_x = read_line_info()[0]
lines_output_points_y = read_line_info()[1]
lines_cycle_times = read_line_info()[2]
speed = 1.4 #[m/s] - Average speed meant to
...ize = 4.8 #kWh
```

```
SYSTEM PERFORMANCE:
Line 0
Actual production [UL]: 54  -  Maximum production [UL]: 64
Total idle time [min]: 150

Line 1
Actual production [UL]: 56  -  Maximum production [UL]: 106
Total idle time [min]: 456

Line 2
Actual production [UL]: 50  -  Maximum production [UL]: 53
Total idle time [min]: 47

Line 3
Actual production [UL]: 55  -  Maximum production [UL]: 80
Total idle time [min]: 300

Line 4
Actual production [UL]: 8  -  Maximum production [UL]: 106
Total idle time [min]: 888
```

Assignment
Part 1 – Tasks

1. Identify the **“weaknesses” of the proposed model (“BASE CASE”)**, **describing them and commenting on their impact on the model quality** (i.e., accuracy in representing the real-life problem and decision-making)

   Examples of “weaknesses”:

   - One tugger train instead of a fleet
   - Weight capacity not modelled
   - …

2. Create an updated version of the model and estimate the **required number of tugger trains** that allows achieving a “reasonably low” idle time (average of the idle times of the 5 production lines lower than 5 minutes during **two 8-hours shifts →** 16 hours)

   N.B.: the updated version is obtained by “solving” one or more of the previously identified weaknesses (you will choose which and how many weaknesses to “solve”)

Assignment
Part 1 – Output

- **Power point** presentation of **15-20 slides**
- **Python script**
- **Excel file (if needed)**

- The files should be **uploaded on WeBeep (*Assignments* Section) in a zip folder named "Group_XX"**, where XX is the group number

Assignment
Evaluation criteria

- **Fulfilment of the requirements (tasks and deadline)**
- **Quality of:**
  - list of weaknesses and related motivations
  - identified number of vehicles
  - updated version of the BASE CASE model
- **Originality**
- **Quality of the .ppt document** (i.e., sequence of the contents, clear description of the work, document readability)

N.B.: after the delivery of the assignment, each student will be asked to fill in a MS Forms to communicate whether all the group members have actively participated

Questions?

For **any additional questions and/or comments** that you might have during the course, you can use the following channels:

- **Email**: emilio.moretti@polimi.it
- **Anonymous feedback** platform: https://freesuggestionbox.com/pub/dhdrzka