CLD771 Minor Project Week Report (05/11/2021)

Weekly Report: 5 November, 2021

Summary of work done

Orders:

I have tested out a few and decided on a system to implement stochastic order generation. This generates orders for the scheduling horizon specified by the environment's `k`.
The specifics of the order system needed to be ironed out before the environment could be finished (as taking a step in the environment invloves details like what order was delayed, what order was delivered, what are the costs (and income) of each order *[required for the reward function]*)

An order 'sheet'/'book' will look something like this:

idx	order_date	due_date	amount	margin	product
0	3	5	100.0	29.5	A
1	7	9	100.0	63.1	A
2	9	10	100.0	33.5	A
3	2	3	100.0	8.4	B
4	4	6	100.0	46.9	D
5	0	0	100.0	58.3	A
6	1	5	100.0	13.3	C
7	4	8	100.0	19.1	D
8	7	10	100.0	55.1	C
9	7	10	100.0	48.6	D

Every order has 5 properties alongside the `index`/`idx`:

1. `order_date`**:** The date on which that order is **placed** (so to the agent, only orders placed upto that date will be visible, for e.g., only orders with `order_date ≤ 2` *[order ids 3, 5, 7, 9]* will be visible to the agent at day 0 of the simulation).
   This is sampled from a discrete uniform distribution from `0` to the last day, i.e. `k`(in our environment)
2. `due_date`**:** The last date by which that order is to be **delivered** without any tardiness costs (after that day tardiness costs will be applied as 25% for each succesive late day).
   This is taken as the `order_date` + a *delay*, sampled from a discrete uniform distribution from `0` to `k/2`. *[*`k\2` *was chosen arbitrarily so that most orders have* `>2` *days in between* `order_date` *and this* `due_date`*]*

3. `amount`: The **amount of product ordered** to be produced. Currently this is a constant.
4. `margin`: The **financial gain** from delivering that order on/before time.
   This is sampled as a number from a standard normal distribution (`Z(mean=0, std=1)`) multiplied by the amount (so that in future even if we change amounts to be unequal, this already takes it into consideration *[since more amount of product ordered usually refers to more money spent]*)
5. `product`**:** This tells us **which product** is ordered out of a variable number of products. *[For the environment the default is 5 products* `A`, `B`, `C`, `D`, `E`*]*.
   This is randomly chosen from all the products. *[i.e., a uniform discrete distribution]*

Agents:

The agents implemented are `DDPG` (Deep Deterministic Policy Gradient) and `PPO` (Proximal Policy Optimization).

I need to implement `A2C` (Advantage Actor-Critic) or `A3C` (Asynchronous A2C) agents.

I'm using PyTorch to implement the neural network part in any of these.

I have additionaly made a graphic demonstrating the basic environment model and flow of components/mixtures (for the final report).