

TODO Title

TODO SubTitle

1 Robo Cup Logistic League Motivation

The Robo Cup Logistics League (RCLL) is designed to represent a common scenario in Industry 4.0. There one needs to assemble products based on dynamic custom orders in a real-time environment. In turn the environment is driven by a consistent stream of orders and individual delivery windows. The challenge here consists in scheduling the assembly process of orders in a feasible manner, to abide the delivery window of as many orders are possible.

2 Reinforcement Learning approach

We present a solution using Reinforcement Learning (RL) to tackle the problem of deciding which assembly step the robots need to process next. Due to the complexity of the problem a straightforward Deep Q-Learning approach is first tried, while keeping open options described in section 3.

We avoid needing a multi-agent model by focusing on the order schedule and deducting a task sequence based on it. Each task t_k will represent a sub-step needed to complete order $O = \{t_1, t_2, \dots, t_n\}$ and will be modelled as $t_k = \mathcal{N}(\mu_k, \sigma_k^2) + \mathcal{U}(a_M, b_M)$. Here the normal distribution \mathcal{N} depends on $\mu_k = d_{ij}$ drawn from a distance matrix representing the distance between machines M_i and M_j . The σ_k can either be fine-tuned by hand or matched from real recorded robot data.

This task sequence can be forwarded

3 Risk mitigation

- If we need to reduce complexity or in contrast intertwine the actual robot movement deeper into the model, we can apply similar multi-skill and multi-machine approaches as described in [\[QWGL16\]](#).
-

3.1 Environment

In the first steps we apply a more simplified environment, which for starters only provides an order schedule with delivery windows. The initial goal is thus for the system to learn which orders are most feasible to complete or award most intermediate rewards.

We build the environment so that we can have a precision of seconds, while not specifically modelling each second as a discrete time step. The idea is to merge the discrete time into the actual state as a numerical feature, which is incremented in the expected passed time for a processing step. Initially the elapsed time for the intermediate steps is drawn from Gaussian distributions, which can be later extended with application of the distance matrix and real-world data.

The environment utilizing most of the complexity behind the task

3.2 States

The state space consists of orders, up to 3 products in the pipeline, distance matrix of the machines, availability of machines and current time. While the distance matrix is constant throughout a game, the other parameters are simulating a dynamic real-time environment.

Specifically we can represent an order as a vector of $O = \{\mathcal{B}, \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{C}, n, c, d_{start}, d_{end}\}$, where

- \mathcal{B} represents a set of base types
- \mathcal{R} is one of up to 3 ring types
- \mathcal{C} is the cap type
- n is the requested amount
- c indicates whether the order is competitive
- d_{start} is the start of the delivery window

- d_{end} is the end of the delivery window

A intermediate product will look like $\{\mathcal{B}, \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3\}$, which corresponds to the physical object one of the 3 robots can transport. We can also omit the cap as we will deliver straight after.

The distance matrix and

While there is still room for changes the current dimensions are

3.3 Actions

The actions

3.4 Rewards

The rewards will be currently

References

- QWGL16. Shuhui Qu, Jie Wang, Shivani Govil, and James O Leckie. Optimized adaptive scheduling of a manufacturing process system with multi-skill workforce and multiple machine types: An ontology-based, multi-agent reinforcement learning approach. *Procedia CIRP*, 57:55–60, 2016.