Mass customization with reinforcement learning: Automatic reconfiguration of a production line

Jifei Deng a,∗, Seppo Sierla a, Jie Sun b,∗, Valeriy Vyatkin a,c

a Department of Electrical Engineering and Automation, School of Electrical Engineering, Aalto University, Espoo, Finland
b State Key Laboratory of Rolling and Automation, Northeastern University, Shenyang, China
c Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå, Sweden

A R T I C L E   I N F O

Article history:
Received 16 April 2022
Received in revised form 9 January 2023
Accepted 10 June 2023
Available online 19 June 2023

Keywords:
Mass customization
Strip rolling
Machine learning
Multi-agent reinforcement learning

A B S T R A C T

This paper addresses the problem of efficient automation system configuration for mass customization in industrial manufacturing. Due to the various demands from customers, production lines need to adjust the process parameters of the machines based on specific quality parameters. Reinforcement learning, which learns from samples, can tackle the problem more efficiently than the currently used methods. Based on the proximal policy optimization and centralized training with decentralized execution, a multi-agent reinforcement learning method (MARL) is proposed to reconfigure process parameters of machines based on the changed specifications. The proposed method has the actor of each agent observing only its own state, the agents are made to collaborate by a centralized critic which observes all the states. To evaluate the method, a steel strip rolling line with six collaborating mills is studied. Simulation results show that the proposed method outperforms the existing methods and state-of-the-art multi-agent reinforcement learning methods in terms of accuracy and computing costs.

© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Mass customization (MC) relates to the ability to provide customized products or services through flexible processes in high volumes and at reasonably low costs [1]. In advanced industrial manufacturing, to meet the customers’ demands, factories are required to flexibly produce products with various specifications [2]. Based on the capability and function of the machines and customers’ demands, control systems need to compute and adjust new process parameters for the machines. In other words, a method is required to reconfigure process parameters of machines based on the changed specifications. When receiving new demands (quality parameters), this method would automatically compute the optimal process parameters for machines.

The methods currently used in industry are mainly based on trial-and-error adjustment of parameters. When building and configuring the control system, engineers and experts in manufacturing and control will manually set the parameters based on their experience. Strips with various specifications will be produced by running the production line, through evaluating the quality parameters of the strips, the process parameters will be adjusted to obtain near-optimal values. The existing method is risky and inefficient, since the attempts of parameters are required, and a large number of strips will be wasted. More advanced methods in academia use evolutionary algorithms (EA), including genetic algorithm (GA) and particle swarm optimization (PSO) [3,4]. The first principle mathematical models are established based on real physical characteristics of the machinery. Based on these characteristics, engineers use EA to find the optimal parameters of the models. However, neither trial-and-error nor EA can adapt well to changes in dynamic conditions. For a new set of product specifications, recomputation of the parameters is required, which is inefficient.

In this paper, we investigate this approach in the example of strip rolling. Steel strips are the raw material for modern industries, including shipbuilding, aerospace, automotive, and defense machine production. The process parameters of the hot rolling mills include rolling force, bending force, and roll shifting, which determine the product quality (thickness and crown). The control system is required to provide setpoints of process parameters for various specifications of strips. However, the computation of the setpoints is difficult. Firstly, the unsteady state of the rolling process results in cross-coupling and changeable constraints, which exponentially increase the difficulty of building a first principles model [5]. Secondly, the occurrences of heating transformation, thermal expansion, abrasion, and vibration of the mills require advanced domain knowledge for modeling [3]. Thirdly, the high-dimensional, nonlinear, and coupled nature of
the problem hinders the first-principles modeling of process parameters and quality parameters. In other words, the computation of process parameters for personalized specifications is a challenging task.

In modern strip rolling factories, expert knowledge and experience combined with trial and error have been applied to build models of the process and quality parameters, by considering the principal potential specifications [3]. A model is built based on the experience of engineers and the number of potential specifications, but its accuracy is hard to guarantee, and manual debugging is inefficient [3]. EA is popular and widely used for optimization problems, also in the strip rolling industry [4, 6]. Given a model (e.g., a set of mathematical equations), and constraints, EA can search and find the optimal parameters for the model [3]. However, the training process is computationally expensive, because the real problem can be changeable, which requires computation for each new condition. In other words, in the strip rolling industry, a time-consuming new computation is required for every new specification.

Since machine learning methods can mathematically extract the underlying properties of the process directly from data, data-driven approaches are developed using the data instead of building physical models. Recent studies show that reinforcement learning (RL) can cope with high-dimensional stochastic problems. During the training process, the agent is not told what to do, but instead must discover which actions yield the most reward by trying them. The systems learn from data and make decisions with minimal human intervention. RL showed feasibility and efficiency in decision-making of real-world problems. In the strip rolling field, RL could learn an optimal policy to give optimal process parameters. Compared to the existing method introduced in paragraph 2 of this section, RL methods can be adaptive to changeable conditions. The reward function which shows the performance of the strips can promise that the policy will be updated in the right direction, making the learning process reasonable for the rolling process.

Another technology that is commonly used in the context of mass customization is multi-agent systems (MAS) [7]. MAS assumes multiple collaborating agents working together to control multiple machines. This architecture helps to reduce the complexity of finding an optimal adaptation of the production system to changing environment conditions, failures, and product specifications.

In [8], the rolling process was modeled as a MAS, and a distributed containment control scheme was proposed for the nonlinear MAS using an adaptive compensation technique. Making use of multi-agent technology, an intelligent strip rolling control system with optimal control agents and diagnostic agents was designed for real-time control [9]. A finishing rolling process of a hot rolling line with six rolling mills was studied. The mills roll the strips one by one to control the strip quality, and each mill has three process parameters. Therefore, the system is naturally defined as a MAS, where each mill is an agent, and the agents collaborate to produce the strips. In this paper, the finishing rolling system is also modeled as a MAS, but then RL is used to define behavior of agents using the multi-agent reinforcement learning (MARL) method. This helps to avoid the manual development of the agents, substituting it by the formal training method based on data. MC means using MARL to compute process parameters of the mills based on the specific quality parameters of strips. Proximal policy optimization (PPO) [10, 11] is a widely used deep policy gradient algorithm, which is adopted as the basic algorithm of MARL in this paper. Using centralized training and decentralized execution (CTDE) [12], PPO is expanded to multi-agent PPO (MAPPO) to solve the studied problem.

The intended contribution of this paper is to develop a novel RL-based approach for reconfiguring a production line based on the changed product specification. The approach aims at the following results:

1. For the first time, a novel mass customization implementation method based on MARL is proposed. MARL is used to build models of process parameters and quality parameters. The method computes the optimal process parameters for the machines based on the quality parameters of products from the customer order.
2. A novel MAPPO based on CTDE is proposed, the centralized critic observes all the states of the actors, guiding cooperation among actors without the need of a communication channel.
3. The efficiency of the proposed method is demonstrated by comparing it with the state-of-the-art MARL methods and existing evolutionary methods in terms of the cumulative rewards and training costs in experiments.

The rest of the paper is organized as follows. Recent approaches for MARL are reviewed in Section 2. Section 3 introduces the studied strip rolling line and Section 4 illustrates the modeling of the system for RL. The proposed method and results are described and analyzed in Sections 5 and 6. Section 7 concludes this paper and describes our future work.
2. Related works

RL is usually referred to single agent RL, given an environment, the agent interacts with the environment to learn a policy [13]. The state-of-the-art single agent RL methods include PPO [10], Twin Delayed DDPG (TD3) [14], and Soft Actor-Critic (SAC) [15]. In the field of finance [16], assembly scheduling problems [17], and industrial mining complexes [18], RL has already achieved great success in control efficiency and energy savings. In MC, Zhou et al. [19] proposed a new RL based cyber–physical integration for scheduling problems of low-volume-high-mix orders generated by mass customization. In [20], interactive RL which learns a complete collaborative assembly process was adapted to reduce programming effort required by an expert using natural modes of communication in robotic systems. To reduce carbon consumption and improve material utilization in printed circuit board manufacturing, Leng et al. [21] designed an order acceptance decision model using RL for orders with a variety of specifications generated by MC.

Compared with single agent RL, in which only one agent interacts with the environment, agents in MAS need to communicate with each other to accomplish the task. According to the relationships between the agents, MARL has three types of methods: fully cooperative, fully competitive, and a mix of the two [22]. For a fully cooperative task, agents collaborate to optimize a common long-term return. In a competitive task, the return of the agents usually sums up to zero. A mix task usually has a general sum return. Currently, fully cooperative tasks are the most common in autonomous vehicles [23], power allocation for 5G [11] and energy management systems [24]. Independent learning (IL), centralized learning are the main training schemes for MAS. Using IL, the most conventional method is to have each agent learn independently while considering other agents as a part of the environment [25]. In IL, each agent learns its own policy independently based on its own observations. This method has been applied in [25,26]. However, the training cost is expensive in terms of computation, and the number of agents is limited. CTDE features centralized learning which is another popular approach to expanding single-agent methods to multi-agent ones [12]. Based on CTDE and deep deterministic policy gradient (DDPG), the multi-agent DDPG (MADDPG) was proposed in [27]. A centralized critic which can observe the states of all the agents is used to evaluate the policies, and decentralized policies, with each agent taking actions based on its local observations, have the advantage of limited communication needs during execution [22]. Using joint action-value factorization techniques is another breakthrough, in 2017, Value-decomposition networks (VDN) was proposed by representing joint action-values as a summation of local action-value conditioned on individual agents’ local observation history [28]. Using a mixing network to approximately a broader class of monotonic functions, QMIX was proposed in 2018 [29]. In 2021, to extend the value-decomposition to actor–critic methods, Value-Decomposition Multi-Agent Actor-Critics (VDAC) was proposed and a balanced trade-off between training efficiency and performance was obtained [30].

Today, EAs have been widely adopted in various fields, and the main EAs include genetic algorithm (GA), genetic programming (GP), differential evolution, and particle swarm optimization (PSO) [31]. In control and optimization fields, EAs have been used to address problems of general parameter control [32], traffic signal control [33], and machine learning parameter optimization [34]. In the strip rolling industry, robust optimization models and the corresponding solutions were proposed using EA to address scheduling problems [35]. Based on GA and linear regression, a hybrid method was proposed for steel strip width deviation prediction [36]. To control the resonant vibration of the rolling mills, a new particle damping absorber was developed and adaptive GA was used for parameter optimization [37]. For MC of strip rolling, since MC is required to meet the demand of strips with various specifications and steel grades by computing process parameters of the rolling mills for each strip, trial-and-error, and EAs have been adopted to the current factories with experts’ knowledge and experience to build the models and tune the parameters.

The research gap of this paper is that the existing methods highly rely on engineers’ experience and knowledge, which means such methods are inefficient and imprecise. Using EA or trial-and-error methods is computationally expensive. New intelligent and flexible methods which can address the limitations of the existing methods are in demand to improve product quality and efficiency. Compared with EA, which has a time-consuming computation process, RL methods which can learn from the data could be a promising tool. In this paper, to address MC problem of strips, and motivated by the distributed structure of the production line which has multiple individual mills, a MARL based method will be trained for the process parameter configuration of various specifications of strips in the strip rolling system. Each rolling mill is modeled as a separate agent, since the mills collaborate to process the strips, a fully cooperative MARL will be studied for MC of the rolling process.

In the real manufacturing industry, supervised learning has been widely applied for predictive maintenance, such as strip crown prediction [38], necking width prediction [39], etc. However, in MC in the industry, using RL for parameter configuration is a new topic, and currently very few publications on this topic can be found. Moreover, the strip rolling process is of a more discrete nature, where the need to reconfigure the machinery may be required for every next strip. This paper studied a novel framework using RL for process parameter reconfiguration of different specifications in strip rolling. The validity of the proposed method for MC in strip rolling could support further research and application in other process industries.

3. Background of industrial case

Figs. 1 and 2 show the studied 2160 mm hot rolling line. As shown in Fig. 1, it has six four-high mills in the finish rolling region; each mill has two work rolls and two backup rolls. Only the work rolls are in contact with the strips. The backup rolls are in contact with work rolls to transfer the rolling force. During the rolling process, to control the thickness and strip crown, two hydraulic cylinders apply a rolling force to the backup rolls, and the work rolls are given a bending force and a roll shifting value.

In the finish rolling process, strips are rolled by the finishing mills in sequence. The process is naturally dynamic, uncertain, and complex. Firstly, process parameters (the rolling force, bending force, and roll shifting of each mill) are determined dynamically according to the real production state. In other words, the production line should be adaptive to various specifications and steel grades to meet the customer’s demand. Secondly, since the finish rolling process includes multiple mills to produce the strips, it is defined as a multi-process production with nonlinear dynamics, where it is difficult to measure its operation status due to the lack of sensors. In practice, modeling the process using mathematical models requires advanced domain knowledge of heating transformation, thermal expansion, abrasion, and vibration. Thirdly, the unsteady state of the rolling process, cross-coupling, and changeable constraints of all the mills also make the strip rolling control system complex, increasing the difficulty of building first principles models. In short, the traditional method requires mathematical equations to formulate the underlying relationship of the factors. However, as introduced above, the
number of factors increases the complexity, making the work time-consuming and cost expensive. The motivation for using RL methods is to reduce the influence of the above three limitations. The data were collected from the real manufacturing process, and the RL method can automatically learn from the data, extracting the underlying information. Compared to the traditional method, the application of RL methods is expected to be more efficient and accurate. This paper aims to propose an RL policy for parameter configuration of MC in strip rolling, which computes the setpoints of process parameters for basic automation system (BAS). As shown in Fig. 3, in the beginning, BAS starts to turn on the mills only after receiving initial process parameters. The real-time control of rolling a strip is based on that, but the RL policy will not involve in the real-time control.

The mills are distributed along the production line and aim to work independently, which is the reason for choosing a MARL method. To produce high quality at a specific sheet thickness and crown, each mill should consider other mills to adjust the process parameters. In other words, a control system is required to make the mills’ control agents collaborate. The finishing rolling area was modeled as a MAS where each mill can be an agent. MARL methods can give adaptive and intelligent policy by learning from the data. CTDE which has a centralized critic to criticize the actors, can flexibly expand the centralized RL methods to MARL.

4. Problem modeling

Accordingly, this paper presents a data-driven optimization method for the process parameters of the hot rolling control system. The proposed architecture is shown in Fig. 3. The scheduling unit determines the target values \( z^* \) of the strip performance indices, and the process control system computes the process parameters \( (u^1, u^2, \ldots, u^N) \) which are also known as setpoints for the basic automation system which controls \( N \) rolling mills. The real performance of the strip is \( z \). According to the structure in Fig. 3, the system can be defined as a MAS, in which each rolling mill is an agent. RL methods will be used to search for the solutions to this MAS.

4.1. Multi-agent task modeling

In this paper, we consider MARL methods that are fully cooperative and completely observable. All the agents are attempting to maximize the discounted sum of the joint reward. The agents cannot explicitly communicate and must learn cooperative behavior only from their observations.

A fully cooperative multi-agent task, in which a team of agents cooperates in a stochastic environment, is described as a game \( G \), which can be defined by a tuple \( G = (S, N, U, r, Z, \gamma) \). The state of the environment is defined as \( s \in S \). \( N \) is the number of agents, \( a_i \) is the \( i \)-th agent. At timestep \( t \), all the agents choose actions simultaneously, yielding the joint action \( u_t : = \{u_{a_i}^t\}_{i=1}^N \in U \). The joint action \( u_t \) is executed in the state \( s_t \), and the next state \( s_{t+1} \sim P(s_t, u_t) \) is drawn from transition function \( P \). The collaborative reward \( r_t : = \{r_{a_i}^t\}_{i=1}^N \) is returned to the agents, and \( \gamma \in [0, 1) \) is the discount factor. Each agent draws an observation \( z_{a_i}^t \in Z \), and \( z_t : = \{z_{a_i}^t\}_{i=1}^N \) denotes the observations of all the agents. In the completely observable setting, \( z_t \) is equal to \( s_t \).

Actors of \( N \) agents are parameterized by \( \theta : = \{\theta_{a_i}^h\}_{i=1}^N \) and critics are parameterized by \( \phi : = \{\phi^h\}_{i=1}^N \), \( \pi : = \{\pi_{a_i}^h\}_{i=1}^N \) is the set of policies. Each agent has an action-observation history \( \tau_{a_i}^t : = (z_{a_i}^0, u_{a_i}^t) \in \mathbb{R}^t \times U \), which conditions a stochastic policy \( \pi_{a_i}^t(\mu^t | \tau_{a_i}^t) \). The action-observation history of all the agents is \( \tau_t : = \{\tau_{a_i}^t\}_{i=1}^N \).
Fig. 3. Proposed architecture of the control system with data-driven optimization. $\bar{u}$ is the output of the controller, $\tilde{u}$ is the actual value.

Table 1
Data description.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rolling force of M1–M6</td>
<td>6</td>
</tr>
<tr>
<td>Bending force of M1–M6</td>
<td>6</td>
</tr>
<tr>
<td>Roll shifting of M1–M6</td>
<td>6</td>
</tr>
<tr>
<td>Thickness ($h$)</td>
<td>1</td>
</tr>
<tr>
<td>Crown ($C$)</td>
<td>1</td>
</tr>
</tbody>
</table>

4.2. Environment

The environment was built based on a dataset recorded at the real production line introduced above, with 2000 data samples collected within a month. The dataset is proprietary and was made available to the authors in a collaborative research project with the enterprise. The data description is shown in Table 1. Each mill has three process parameters (rolling force, bending force, and roll shifting), which were collected from the controllers. The environment was built based on a neural network with two hidden layers. For the environment network, 18 process parameters are the inputs, and thickness and crown are the outputs. All the data were collected when the production process was operated under normal conditions. Multi-function gauges were used to measure the thickness and strip crown. The data were standardized by removing the mean and scaling to unit variance. Mean squared error and Adam [40] were used to optimize to network. Following the framework of Gym from OpenAI [41], the network was embedded in the “step” function. Moreover, a “reset” function was set to generate initial states, and a “reward” function was used to calculate the reward which was used as an indicator of the training performance. At timestep $t$, the environment in Fig. 4 returns a joint observation $z_t$ and a reward $r_t$ after receiving a joint action $u_t$. The joint action $u_t$ consists of the action of all the agents $u_t = \{u_{at}\}_{a=1}^{N}$, in this paper $N = 6$. Like joint action, joint observation $z_t$ consists of the observations of all the agents $z_t = \{z_{at}\}_{a=1}^{N}$.

4.3. Action

Each mill has three continuous process parameters described in Table 2, including rolling force ($FR$), bending force ($FB$) and roll shifting ($DS$). For this case, these three parameters are the actions of each agent. In other words, each agent has three continuous actions. At timestep $t$, the action of agent $a$ is $u_{at} = \{FR_a, FB_a, DS_a\}$. Moreover, safety is guaranteed by manually specifying constraints on the policy’s behavior. The actions will be adjusted to the maximum/minimum when the policy gives the values out of the range described in Table 2.

4.4. Observation

The thickness ($h$) and strip crown ($C$) are used to indicate the shape of the strips. At timestep $t$, observation of each agent is $z_{at} = \{C_{at}, h_{at}\}$. The strip crown is defined as:

$$C = h_c - \frac{h_{\text{left}} + h_{\text{right}}}{2}$$

(1)

where $h_c$ is the thickness at the center of the strip, $h_{\text{left}}$ and $h_{\text{right}}$ are the thickness at 40 mm on the left and right sides, respectively. The observed thickness ($h$) is the average thickness of the section measured by the thickness sensor.

4.5. Reward

The purpose of the control system is to give the optimal process parameters for the mills, producing the strips with the final thickness and crown close to the target values. The reward

Fig. 4. The structure of the environment of strip rolling.

Table 2
Process parameters.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rolling force ($FR$)</td>
<td>0–45000 kN</td>
</tr>
<tr>
<td>Bending force ($FB$)</td>
<td>0–2000 kN</td>
</tr>
<tr>
<td>Roll shifting ($DS$)</td>
<td>–150–150 mm</td>
</tr>
<tr>
<td>Thickness ($h$)</td>
<td>2.5–6 mm</td>
</tr>
<tr>
<td>Crown ($C$)</td>
<td>20–75 µm</td>
</tr>
</tbody>
</table>
function $r$ is defined as the absolute error between the observed and target values.

$$r = \frac{1}{N} \sum_{i=1}^{N} (-0.1 \cdot |C^i - C^{\text{tar}}| - 10 \cdot |h^i - h^{\text{tar}}|)$$  \hspace{1cm} \text{(2)}$$

where $C^{\text{tar}}$ and $h^{\text{tar}}$ are the target crown and thickness, $C^i$ and $h^i$ are the output crown and thickness. The joint reward is the average reward of all the agents. The parameters $-0.1$ and $10$ are selected based on the experience of the experts working at the studied production line. This expertise is specific to the studied problem. Because the model is more sensitive to the crown, given different inputs, the crown changes in a big interval, while the thickness changes in a small one. If thickness and crown contribute to the reward equally, the changes in thickness could be ignored. Therefore, absolute errors are scaled to make the crown and thickness equally contribute to the reward.

### 5. Proposed method

#### 5.1. Proximal policy optimization

Policy gradient methods compute an estimator of the policy gradient and plug it into a stochastic ascent algorithm. Policy gradient estimates can have high variance \cite{42} and algorithms can be sensitive to the hyperparameters. PPO relies on specialized clipping in the objective function to remove incentives for the new policy to get far from the old policy. It has an actor-critic framework, which has a policy and a value function \cite{10}. With supervised learning, it is easy to implement the cost function, run gradient descent on it, and get excellent results with relatively little hyperparameter tuning \cite{10}. This characteristic makes PPO better for industrial cases. In PPO, the policy network is trained with the clipped surrogate objective:

$$L^{\text{clip}}(\theta) = \min \left[ \pi_\theta (u|s), \text{clip} \left( \pi_\theta (u|s), 1 - \epsilon, 1 + \epsilon \right) \pi_{\text{old}} (u|s) \right]$$  \hspace{1cm} \text{(3)}$$

where $\pi_\theta$ is a stochastic policy parameterized by $\theta$, $\epsilon$ is a hyperparameter, $\epsilon = 0.2$, and $A_t$ is an estimator of advantage function at timestep $t$. $A_t$ is an estimator of the advantage function, $u_t$ and $s_t$ are the action and the state at timestep $t$. $A_t$ is computed with generalized advantage estimation (GAE) \cite{43}.

Considering an entropy bonus $\hat{S} [\pi_\theta]$, two terms can be combined to the following objective:

$$L^{\text{clip} + \hat{S}}(\theta) = \hat{S} \left[ L^{\text{clip}}(\theta) + cS [\pi_\theta] \right]$$  \hspace{1cm} \text{(4)}$$

where $c$ is the hyperparameter for entropy, and $c = 0.01$ in this paper. To train the value function network, the objective is given by:

To train the value function network, gradient descent is used to minimize

$$L^{\text{value}}(\phi) = \hat{S} \left[ \left( V_\phi (s_t) - R_t \right)^2 / 2 \right]$$  \hspace{1cm} \text{(5)}$$

where $V_\phi$ is the state-value function parameterized by $\phi$ and $R$ is the return.

#### 5.2. Proximal policy optimization for multi-agent systems

For the MAS of strip rolling, based on the PPO, CTDE was used to make the agents collaborate. With CTDE, each actor conditions only on the agent’s own action-observation history. The CTDE uses centralized information to facilitate the training of decentralized policies \cite{12}. In Fig. 5, for agent $a_i$, it has an actor $\pi^i (u^i|s^i)$ and a critic $V (s^i)$. The centralized critic that has access to all the observations is used by all agents. The training framework of MAPPO is shown in Fig. 6. Algorithm 1 shows the core MAPPO algorithm in pseudo-code. Meanwhile, DDPG, SAC, and TD3 will be expanded to MADDPG, MASAC, and MATD3 with CTDE framework.

![Fig. 6. The framework of multi-agent PPO.](image)

**Algorithm 1: MAPPO**

```
for episode=1, 2, ..., $E_0$ do
  for $r$ to max-timesteps-per-episode do
    for $i=1, 2, ..., N$ do
      Run policy $\pi^i$ for $T$ timesteps, collecting $(s^i_0, a^i_0, r^i_0)$
      Estimate advantages $A^i_0$
    end for
    for $i=1, 2, ..., N$ do
      for epoch=1, 2, ..., $E_i$ do
        Optimize $L^{\theta^i}_c$ w.r.t $\theta^i$
        end for
      end for
      Update $\theta^i \leftarrow \theta^i + \delta^i$ and $A^i_t \leftarrow \phi^i$
    end for
  end for
```

5.3. Implementation process

In this paper, six multi-agent methods are implemented, including MADDPG, MASAC, MATD3, MAPPO, VDAC, and IPPO, all the critics are only used for training. Only the actors are allowed to receive observations at the test time. All the methods will be trained for 3000 episodes, and each episode gives the agents no more than 500 timesteps to interact with the environment. For MAPPO, every 300 timesteps during the training process, the policies will be updated for 10 epochs. The advantage is normalized by $A = (A - \mu) / \sigma$, where $\mu$ and $\sigma$ are the mean and standard deviation of $A$. Since all the methods are based on the actor-critic framework, they have sharing parameters such as episodes, optimizer, learning rate, and each method has unique parameters, such as entropy regularization coefficient (MASAC), $\lambda$, clip range (MAPPO). The difficulty of parameter tuning is each method has
Table 3

Hyperparameters

<table>
<thead>
<tr>
<th>Hyperparameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodes</td>
<td>3000</td>
</tr>
<tr>
<td>Max timesteps per episode</td>
<td>500</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Learning rate for the actor</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td>Learning rate for the critic</td>
<td>$1 \times 10^{-3}$</td>
</tr>
<tr>
<td>Hidden nodes of the actor and the critic</td>
<td>64</td>
</tr>
<tr>
<td>Activation function</td>
<td>ReLU</td>
</tr>
<tr>
<td>Update every</td>
<td>500</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.99</td>
</tr>
<tr>
<td>MAPPO clip range ($\epsilon$)</td>
<td>0.2</td>
</tr>
<tr>
<td>Optimizer (VDAC)</td>
<td>RMSprop</td>
</tr>
<tr>
<td>$\lambda$ (MAPPO)</td>
<td>0.98</td>
</tr>
<tr>
<td>Minibatches per epoch (MAPPO)</td>
<td>60</td>
</tr>
<tr>
<td>Update epochs (MAPPO)</td>
<td>10</td>
</tr>
<tr>
<td>STD for Gaussian noise added to policy (MADDPG, MATD3)</td>
<td>0.1</td>
</tr>
<tr>
<td>Start steps (MADDPG, MASAC, MATD3)</td>
<td>5000</td>
</tr>
<tr>
<td>Update after (MADDPG, MASAC, MATD3)</td>
<td>3000</td>
</tr>
<tr>
<td>Batch size (MADDPG, MASAC, MATD3)</td>
<td>100</td>
</tr>
<tr>
<td>Update epochs (MADDPG, MASAC, MATD3)</td>
<td>50</td>
</tr>
<tr>
<td>Polyak (MADDPG, MASAC, MATD3)</td>
<td>0.995</td>
</tr>
<tr>
<td>Entropy regularization coefficient (MASAC)</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Fig. 7. Sensitivity of MASAC to Entropy regularization coefficient.

Fig. 8. Comparison of MADDPG, MASAC, MATD3, MAPPO, and VDAC in terms of cumulative reward.

6. Results

6.1. Comparison analysis

The training process of five methods is shown in Figs. 8 and 9. The cumulative reward of each episode was recorded, and all the methods can converge to stable values using no more than 500 episodes. Shadow areas in Figs. 8 and 9 are based on the mean and the standard deviation, which are computed by repeating the training. As shown in Fig. 8, MADDPG (green line) and MATD3 (orange line) have shadow areas beyond the optimal reward 0, which means the rewards of different repetition training vary dramatically, resulting in a large standard deviation. In statistics, the mean plus standard deviation can be larger than the maximum (the optimal reward).

However, MASAC (red line) has a stable cumulative reward of around $-75$ which is much lower than others. It means MASAC can also solve this problem, but the lower reward means for each episode, MASAC needs more timesteps to solve the problem, less efficient than other methods. For MADDPG, MATD3, VDAC, and MAPPO, all of them can converge to stable values which are close to 0.

Fig. 9 shows the training performance after the five methods converge to stable values. First, MAPPO has the highest cumulative reward (blue line) which is close to 0 after being trained for 3000 episodes. However, MADDPG and MATD3 required only about 100 episodes to converge, while MAPPO used about 750 episodes. It means MAPPO was slower to learn a policy than MADDPG and MATD3. For VDAC, it used around 700 episodes to converge, which is more efficient than MAPPO, but the optimal reward ($-5.898$) is lower than that of MAPPO from Fig. 9 and Table 4. Each method was run 5 times, outputs of VDAC and MAPPO were stable. MADDPG and MATD3 have been accompanied by the exploration process in the later stage, causing the reward to fluctuate widely. In other words, MAPPO performed more smoothly than MADDPG and MATD3 according to the shadow area. It can be concluded that, in terms of cumulative reward,
Fig. 9. Comparison of MADDPG, MASAC, MATD3, MAPPO, and VDAC in terms of cumulative reward over $-200$.

Table 4 Optimal mean reward of each method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimal reward</th>
<th>Episode obtained at</th>
</tr>
</thead>
<tbody>
<tr>
<td>MADDPG</td>
<td>$-1.934$</td>
<td>2266</td>
</tr>
<tr>
<td>MASAC</td>
<td>$-33.060$</td>
<td>1925</td>
</tr>
<tr>
<td>MATD3</td>
<td>$-2.742$</td>
<td>1677</td>
</tr>
<tr>
<td>MAPPO</td>
<td>$-0.888$</td>
<td>2958</td>
</tr>
<tr>
<td>VDAC</td>
<td>$-5.898$</td>
<td>2942</td>
</tr>
<tr>
<td>IPPO</td>
<td>$-418.057$</td>
<td>2643</td>
</tr>
</tbody>
</table>

MAPPO are the most stable, and has the best performance. The optimal mean reward and the episode obtained at are shown in Table 4, and the optimal policy was recorded for evaluation (discussed in Section 6.2). Figs. 13 and 14 were generated by the recorded optimal policy.

In addition to the cumulative reward, computing times of 3000 episodes were recorded and compared. After repeating the training 5 times, the average value and STD of computing time are listed in Table 5. First, the average computing time for each training of MASAC is 2828.68 s, much higher than MADDPG (326.83 s), MATD3 (541.13 s), MAPPO (294.78 s), and VDAC (182.86 s). MADDPG, VDAC, and MAPPO have lower training costs than MATD3, and the STD of MATD3 is 178.90 s which is higher than MADDPG (29.55 s), VDAC (46.80 s), and MAPPO (35.99 s). Although the STD of MADDPG is lower than that of MAPPO, the average value of MAPPO (294.78 s) is much lower.

Moreover, Fig. 10 compares the MAPPO with independent PPO (IPPO). For IPPO, each agent has a separate actor-critic network, which can only observe its own states. IPPO can converge to around $-500$, which requires over 1000 episodes, but MAPPO only requires 750 episodes to converge to 0. The lines in the figure show that MAPPO is much more stable than IPPO. With a multi-agent framework, MAPPO outperforms IPPO in terms of stability and cumulative reward.

Based on the above analysis, MAPPO outperformed all the other baselines in most metrics, it can tackle the strip rolling problem with accuracy and efficiency. The model training is implemented by using Python 3.8 on Ubuntu 20.04, with Intel Core i7 @ 2.30 GHz×4, Nvidia Quadro RTX 5000, and 6 GB memory.

Fig. 10. Comparison of IPPO and MAPPO in terms of cumulative reward.

6.2. Analysis of strip rolling control

According to the comparison analysis, MAPPO is the optimal RL method for the studied problem. In this section, conventional methods with GA and PSO were studied to search parameters for the models. Mean squared error was used as the fitness function. The training process of GA is shown in Fig. 11, including the mean fitness function value (a) and best fitness function value (b). Due to the population size, although the best value at the first generation was 35, the mean value was over 4000. During the training process, either mean values or best values were decreasing, and the mean values were approaching the best values. After being trained for 1000 generations, GA converged, the mean fitness value was 0.353, and the best fitness value was 0.323.

The training process of PSO is shown in Fig. 12. Similar to GA, due to the swarm size, mean function values were much higher than the best fitness values at the beginning. The training made PSO converge, from the 60th to the 100th iteration, the mean values were close to the best values. For PSO, the best fitness function value was 0.611.

According to the above analysis, GA has the best performance, since it obtained a lower fitness function value. GA took 1000 generations to converge and PSO took only 100. To evaluate the computational efficiency, the computing time was recorded in Table 5, the average computing time of GA is 300.69 s, and PSO takes 188.37 s. Although GA took more time (112.32 s) than PSO, the computing cost of either GA or PSO was low and acceptable. In terms of best function value and computational efficiency, GA outperformed PSO, which will be chosen as the baseline of conventional EA algorithms. The hyperparameter settings of GA and PSO are shown in Tables 6 and 7.

In Figs. 13 and 14, GA is the existing method and RL is the MAPPO method. First, given a steel grade, a target specification...
is fed to the models (GA and RL) separately. Then, the models will compute and output the actions for the machines, feeding the actions to the environment separately. Finally, the environment outputs new states (specifications). The model with higher performance will have the new specification closer to the target one. Fig. 13 shows the absolute error of the thickness (a) and strip crown (b) achieved with GA and RL methods. Although either GA or RL has a smooth line in thickness, the values of absolute error generated by RL (around 0.01) are lower than that of GA (around 0.15). For the strip crown, as shown in Fig. 13(b), RL method has more points closer to 0, and the values are in a small range of 6 to 7. The conclusion is the RL method outperforms the GA method. Moreover, the actions of the first mill calculated by GA and RL are shown in Fig. 14.

To statistically evaluate the performance of GA and RL, the mean and STD of thickness and crown were calculated. As shown in Table 8, the average thickness error generated by RL is 0.0076, while GA has a higher value of 0.1431. For the strip crown, the average error of RL is 6.4846, lower than 8.4532 which was generated by GA. From Table 8, the statistical results showed that RL method can produce strips with higher quality, where both thickness and strip crown are close to the target values.

### 6.3. Discussion

In this paper, RL was adopted to address MC problem of strip rolling. Considering the safety problem, it is risky to implement...
the training process on the real system, based on the real data, an approximation model (environment) of the system was built for training. Our work demonstrated that RL can be a promising tooling to address such problem, however, a reality gap caused by the approximation model needs to be further studied. Since the model (environment) was built with simplification and approximation, it cannot fully represent the real system, and the deployment of the solution trained on such model could be risky.

To address the gap, we made two future directions. First, work on the model-free MARL, but build a high-fidelity digital twin of the studied process which can guarantee a safer and more reliable solution for deployment. Second, switch to offline RL methods, which can directly learn from the real data rather than the approximation model. Since the bias between the model and the real process is reduced, such method can narrow the reality gap.

7. Conclusion

This paper proposed a novel data-driven method for MC of an industrial production line. MARL method was trained to adjust the process parameters of machines based on the quality parameters of products from customer’s orders. A real strip rolling line was studied, and modeled as a fully cooperative multi-agent task, in which the actions of the agents are continuous. Using the CTDE architecture, a MAPPO method was developed, in which global information was used to train the decentralized policies. Comparison results showed that this method can tackle the studied problem in terms of cumulative reward and computing cost better than state-of-the-art multi-agent methods and existing evolutionary methods.

It is the first paper that studied MC problems in the strip rolling industry using RL method. The results demonstrated that RL based methods could be a promising tool for MC problems of various process industries. Since the current work studied the thickness and crown of hot rolled strips, using transfer learning, our future work will focus on developing a general framework for both hot rolled and cold rolled strips (flatness-related MC). After that, to improve the generalization of the current method, a more general framework for various process industries will be studied by extracting the characteristics in common. Moreover, considering the practical usability, cloud computing and 5G techniques could be adopted, the model can be trained on a cloud computing platform using a huge amount of real production data, then be embedded in local systems through 5G.

CRediT authorship contribution statement

Jifei Deng: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft. Seppo Sierla: Formal analysis, Validation, Writing – review & editing. Jie Sun: Data curation, Writing – review & editing. Valeriy Vyatkin: Formal analysis, Investigation, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.
Acknowledgments

This work was supported by China Scholarship Council (No. 202006080008), National Natural Science Foundation of China (Grant Nos. 52074085 and U21A20117), the Fundamental Research Funds for the Central Universities (Grant No. N2004010), and the LiaoNing Revitalization Talents Program (LXYC1907065).

References