Karafan
Quarterly Scientific Journal of National University of Skill
Spring 2026, Vol. 23, No. 1, p. 362-381
Journal Homepage: https://karafan.tvu.ac.ir/?lang=en
:10.48301/kssa.2025.511129.3166

Solution for Reducing Complexity and Increasing Efficiency in AI-Based Collaborative Robots

Seyed Reza Mazlooman[1], Saeed Setayeshi[2*], Mohsen Jahanshahi [3]

[1, 3] Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran.

[2] Department of Medical Radiation Engineering, Amirkabir University of Technology, Tehran, Iran.

ARTICLE INFO

Article Type:
Original Research

Received: 11.03.2025
Revised: 01.09.2025
Accepted: 15.10.2025

Keyword:
Job Shop Scheduling Problem
Cobot, Markov Decision Process
Multi-Agent Reinforcement Learning
Intrinsic Motivation-Based Approach

***Corresponding Author:**
Saeed Setayeshi
Email: *setayesh@aut.ac.ir*

ABSTRACT

Significant advancements have been made in the development of processing tasks across various industries to enhance productivity, reduce costs, and improve flexibility. However, in the manufacturing industry, resource constraints remain a persistent challenge, often manifesting as precedence relationships among factors. Therefore, task scheduling requires greater precision and adaptability. To address this issue, this study presents a scheduling algorithm that utilizes a multi-agent system to solve the Flexible Job Shop Scheduling Problem (FJSP) in a dynamic input environment. The resource precedence environment is modeled as a decentralized partially observable Markov decision process (Dec-POMDP), where tasks act as independent agents and select an available collaborative robot (cobot) based on their current observations. To overcome the challenge of managing the high-dimensional operational space in concurrent multi-task scheduling, an architecture is designed within a multi-agent reinforcement learning (MARL) system with intrinsic motivation. This model simultaneously considers both coordinated behavior among agents and individual agent performance. The novelty of this research lies in the application of an intrinsic motivation-based multi-agent system for problem-solving and optimizing overall job execution time. Compared to well-known methods such as rule-based approaches, metaheuristics, and cooperative strategies, the proposed model demonstrates superior performance. The results from the case study highlight the flexibility and training stability of this model, suggesting that it can serve as an effective solution for addressing task scheduling challenges in various industries.

E-ISSN: 2538-4430
ISSN: 2382-9796

EXTENDED ABSTRACT

Introduction

In smart manufacturing, robots such as cell robots and collaborative robots (cobots) play a crucial role in logistics, supply chains, and flexible production environments. However, due to limited resources like robots, machines, and tools, tasks cannot always be executed simultaneously. Efficient scheduling is therefore essential for optimizing production processes, resource allocation, and job sequencing, particularly in modern manufacturing systems like cloud manufacturing, mass personalization, and high-mix-low-volume (HMLV) production. One major challenge in smart manufacturing is the Flexible Job Shop Scheduling Problem (FJSP), which is classified as NP-hard, making it computationally complex to solve optimally. Traditional approaches, including meta-heuristics and dispatching rules, perform well under static conditions but struggle with dynamic job arrivals and diverse requirements. Real-time decision-making is necessary to address these challenges. To tackle this issue, we propose a multi-agent scheduling framework for HMLV environments that adapts to dynamically arriving jobs. Various scheduling methods, including meta-heuristics, single-agent models, multi-agent systems, and deep reinforcement learning (DRL), have been explored. However, many rely on assumptions like fixed transportation times, system centralization, or simplified job conditions, which limit their effectiveness. Our approach seeks to overcome these limitations by developing a multi-agent reinforcement learning (MARL) system that minimizes makespan while optimizing cobot and resource utilization.

The proposed system models a flexible job shop environment with multiple workstations, each containing multiple machines capable of performing specific operations. Jobs arrive according to a stochastic distribution, and multiple robots transport them between workstations. The scheduling model incorporates preemptive job execution, allowing higher-priority jobs to interrupt and reschedule lower-priority ones. To manage this complexity, we integrate a preemptive decision-making policy within our MARL framework, enabling job agents to learn optimal scheduling strategies while adapting to unexpected changes. This problem is inherently decentralized, as jobs act as independent agents making scheduling decisions based on system conditions. Our approach employs MARL to optimize both individual job decision-making and overall system coordination. Specifically, we introduce a Multi-agent Scheduling System (MASS) with three key contributions:

i) Intrinsic motivation mechanisms to handle job priority challenges in DRL.
ii) A novel scheduling model via Centralized Training and Decentralized Execution (CTDE).
iii) A dynamic FJSP incorporating stochastic job arrivals and adaptive scheduling policies.

The remainder of this paper covers related work, problem formulation, system architecture, scheduling algorithms, experimental results, and conclusions.

Methodology

We represent the resource preemption environment as a decentralized partially observable Markov Decision Process (MDP), where individual jobs act as agents that select an available cobot based on their current observations. To tackle the challenge of managing a high-dimensional action space in multi-task simultaneous scheduling, we develop an architecture within a multi-agent reinforcement learning system that is driven by intrinsic motivation. This model considers both agent coordination and individual agent behavior. To evaluate the effectiveness of our model, we simulate various FJSP scenarios. We use makespan as the primary performance indicator and compare the performance of our MASS approach with other scheduling algorithms such as PDR, HHA, and MADRL. We conduct comparative experiments on total makespan and training stability, and perform scalability experiments by varying the number of job agents. Detailed scheduling results are presented through Gantt charts. The development platform is based on Python 2.7.3 and PyTorch 1.4.1, running on a single NVIDIA GeForce RTX 2080Ti with 16GB of memory and an Intel Core i7-9700KF CPU. The environment is developed using the OpenAI gym package, which provides interfaces and specifications for RL environments. The unit of time consumption mentioned in this section is minutes.

Results and discussion

We compare the performance of our MASS against several established algorithms: QMIX, a simulated annealing-based hyper-heuristic (SA-HH) method, and two RBSM methods including SPT and earliest due date (EDD), which are widely used PDRs in real-world production settings. In reinforcement learning-based scheduling algorithms, training stability is crucial for ensuring reproducibility and reliability. QMIX is known for its effectiveness in multi-agent systems with a small number of agents. Conversely, MASS is tailored for decentralized MARL environments, where each agent has partial observability and learns from its own experiences while coordinating with other agents. Thus, MASS can be more efficient in settings with a larger number of agents and more complex tasks. To evaluate the robustness of the MASS model, we compare the training processes of MASS and QMIX using three experimental instances (D01-D10). Figure 2 illustrates the total makespan achieved by MASS within the first 8000 training steps, highlighting improved agent coordination and effective internal motivation, which helps avoid delays. The results suggest that MASS efficiently adapts to various situations by learning appropriate behaviors. The intrinsic reward mechanism enhances policy learning: agents receive higher intrinsic rewards for valid policies and lower rewards for faulty ones. The plots Figures 1 depict total rewards. Agents strive to make optimal decisions to maximize rewards, progressing towards both individual and collaborative goals. The results indicate that agents have learned to prioritize tasks based on arrival rates. Further analysis reveals how the attention mechanism aids intrinsic reward learning.

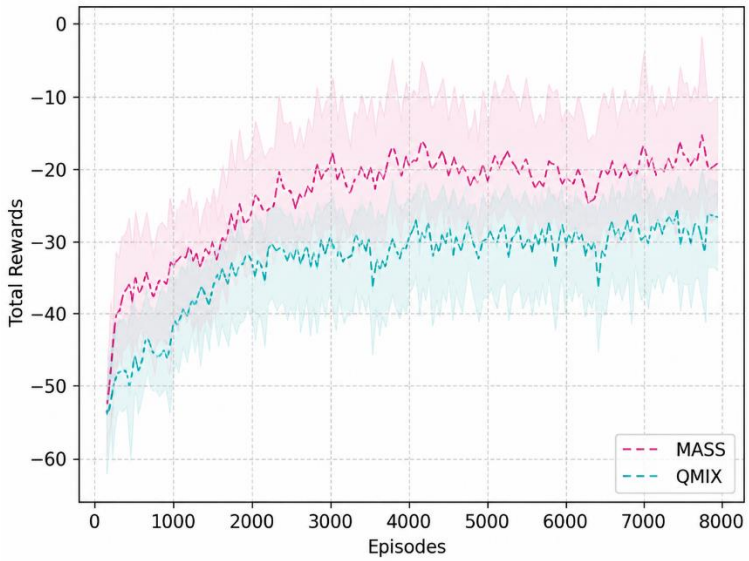


Figure 2. Curve training instance 11, training compare between MASS and QMIX.

Conclusion

This paper presents a Multi-Agent Reinforcement Learning solution based on intrinsic motivation for dynamic Flexible Job Shop Scheduling Problems. We propose a scheduling system that incorporates centralized training, facilitating decentralized scheduling execution. Our approach addresses job scheduling in a multi-agent environment through a decentralized MDP that balances agent coordination and individual behavior. Our intrinsic motivation-oriented multi-agent system outperforms well-known methods in optimizing makespan and can manage dynamic job arrivals under a MARL framework. The study demonstrates the flexibility and training stability of our model, suggesting its suitability for various industrial job scheduling challenges.

کارافن
فصلنامه علمی دانشگاه ملی مهارت
بهار ۱۴۰۵، دوره ۲۳، شماره ۱، ۳۸۱-۳۶۲
آدرس نشریه: https://karafan.tvu.ac.ir
doi :10.48301/kssa.2025.511129.3166

راهکاری برای کاهش پیچیدگی و افزایش بهره‌وری از روبات‌های همکار مبتنی بر هوش مصنوعی

سید رضا مظلومان[۱]، سعید ستایشی[۲*]، محسن جهانشاهی[۳]

۱و۳- گروه مهندسی کامپیوتر، واحد تهران مرکزی، دانشگاه آزاد اسلامی، تهران، ایران.

۲- گروه مهندسی انرژی و فیزیک، دانشگاه صنعتی امیرکبیر، تهران، ایران.

اطلاعات مقاله

نوع مقاله: مقاله پژوهشی

دریافت مقاله: ۱۴۰/۱۲/۲۱
بازنگری مقاله: ۱۴۰۴/۰۶/۱۰
پذیرش مقاله: ۱۴۰۴/۰۷/۲۳

کلید واژگان:
مسئله زمان‌بندی کارگاهی کوبوت
فرآیند تصمیم‌گیری مارکوف
یادگیری تقویتی چندعاملی
محوریت بر انگیزه درونی

***نویسنده مسئول:** سعید ستایشی
پست الکترونیکی:
setayesh@aut.ac.ir

چکیده

پیشرفت‌های قابل توجهی در توسعه وظایف پردازشی در صنایع مختلف به منظور افزایش بهره‌وری، کاهش هزینه‌ها و بهبود انعطاف‌پذیری صورت گرفته است. با این حال، در صنعت تولید، محدودیت منابع به طور معمول یک چالش پایدار محسوب می شود که به شکل روابط تقدم و تأخر میان عوامل نمود پیدا می‌کند. بنابراین، زمان‌بندی وظایف نیازمند دقت و انعطاف‌پذیری بیشتری است. برای رفع این مسئله، این مطالعه یک الگوریتم زمان‌بندی ارائه می‌دهد که از یک سیستم چندعاملی برای حل مسئله زمان‌بندی کارگاهی انعطاف‌پذیر در محیطی با ورودی پویا استفاده می‌کند. محیط تقدم منابع به صورت یک فرآیند تصمیم‌گیری مارکوف غیرمتمرکز و قابل مشاهده جزئی مدل سازی شده است، که در آن وظایف به عنوان عوامل جداگانه عمل کرده و بر اساس مشاهدات فعلی خود، یک همکار روباتیک در دسترس را انتخاب می‌کنند. برای غلبه بر چالش مدیریت فضای عملیاتی با ابعاد بالا در زمان‌بندی همزمان چندوظیفه‌ای، یک معماری در یک سیستم یادگیری تقویتی چندعاملی با انگیزه درونی طراحی شده است. این مدل هم‌زمان رفتار هماهنگ بین عوامل و عملکرد فردی هر عامل را در نظر می‌گیرد. نوآوری این پژوهش در استفاده از سیستم چندعاملی مبتنی بر انگیزه درونی برای حل مسئله و بهینه‌سازی زمان کل اجرای کارها است، که نسبت به روش‌های شناخته‌شده‌ای مانند روش‌های مبتنی بر قوانین، متاهیوریستیک و روش‌های تعاونی عملکرد بهتری دارد. نتایج به‌دست‌آمده از مطالعه موردی نشان‌دهنده انعطاف‌پذیری و پایداری آموزشی این مدل بوده و پیشنهاد می‌کند که این روش می‌تواند راه‌حل مناسبی برای مقابله با چالش‌های زمان‌بندی وظایف در صنایع مختلف باشد.

شاپای الکترونیکی: ۲۵۳۸-۴۴۳۰
شاپای چاپی: ۲۳۸۲-۹۷۹۶

۱. مقدمه

در سال‌های اخیر، با پیشرفت چشمگیر فناوری‌های نوین، تولید هوشمند به یکی از ارکان اساسی صنایع پیشرفته تبدیل شده است. یکی از مهم‌ترین پیشرفت‌ها در این حوزه، توسعه و کاربرد گسترده روبات‌های سلولی و همکار[1] است [۱] که نقش مهمی در بهبود بهره‌وری، کاهش هزینه‌ها و افزایش انعطاف‌پذیری فرآیندهای تولید ایفا می‌کنند [۲]. این روبات‌ها علاوه بر انجام وظایف پیچیده مانند جابجایی مواد و پردازش عملیات، توانایی تطبیق با محیط‌های پویا را دارند [۳]. با وجود این پیشرفت‌ها، مدیریت زمان‌بندی وظایف همچنان چالشی اساسی در صنایع تولیدی محسوب می‌شود. محدودیت منابع نظیر ماشین‌آلات و ابزارها باعث می‌شود انجام هم‌زمان تمامی وظایف ممکن نباشد [۴]. از این رو، زمان‌بندی مؤثر به‌عنوان عاملی کلیدی برای بهینه سازی فرآیندهای تولید و تخصیص منابع مطرح شده است. مسئله زمان‌بندی کارگاهی انعطاف‌پذیر(FJSP)[2] یکی از چالش‌های رایج در این زمینه است که به دلیل پیچیدگی‌های ترکیبی خود به‌عنوان یک مسئله NP-سخت شناخته می‌شود [۵]. روش‌های سنتی نظیر الگوریتم‌های متاهیوریستیک و قوانین اعزام، در شرایط ایستا نتایج نسبتاً مطلوبی ارائه می‌دهند، اما در مواجهه با محیط‌های پویا و وظایف با ورود تصادفی عملکرد ضعیفی دارند [۶]. در این مقاله، مدلی مبتنی بر یادگیری تقویتی چندعاملی[3] ارائه می‌شود که با استفاده از انگیزه درونی، عملکردی مؤثر در حل مسئله FJSP دارد. این مدل به‌گونه‌ای طراحی شده است که عامل‌ها بتوانند به صورت غیرمتمرکز عمل کرده، از تجربیات خود یاد بگیرند و وظایف را بهینه سازی کنند. هدف اصلی مدل پیشنهادی، کاهش زمان کل اجرای وظایف[4] و افزایش بهره‌وری سیستم است.

۲.پیشینه پژوهش

مسئله زمان‌بندی کارگاهی انعطاف‌پذیر یکی از موضوعات اساسی و چالش‌برانگیز در حوزه تولید هوشمند است که توجه بسیاری از محققان را به خود جلب کرده است [۷]. در طول سال‌های اخیر، روش‌های مختلفی برای حل این مسئله ارائه شده است که هرکدام مزایا و معایب خاص خود را دارند. روش‌های بهینه سازی عملیاتی نظیر برنامه‌ریزی خطی و پویا در مسائل کوچک عملکرد خوبی نشان داده‌اند، اما در مسائل بزرگ‌تر و شرایط پویا، کارایی خود را از دست می‌دهند [۸]. از سوی دیگر، روش‌های مبتنی بر قوانین۵، مانند قوانین اعزام وظایف، به دلیل سادگی و سرعت، در بسیاری از محیط‌های صنعتی به کار گرفته می شوند. این روش‌ها شامل الگوریتم‌هایی نظیر کمترین زمان پردازش و اولویت‌بندی بر اساس مهلت انجام کار هستند. با این حال، این روش‌ها در مدیریت محیط‌های پویا و شرایطی که اختلالات مکرر رخ می‌دهند، محدودیت‌های جدی دارند [۹]. الگوریتم‌های ابر-اکتشافی۶، مانند الگوریتم‌های ژنتیک [٤]، سیستم‌های ایمنی [۱۰] و بهینه سازی کلونی مورچگان [۱۱]، توانایی ارائه راه‌حل‌های نزدیک به بهینه را دارند. این روش‌ها با جست‌وجوی فضای راه‌حل‌ها، نتایج قابل قبولی در مسائل ایستا ارائه می‌دهند. با این وجود، به دلیل عدم انعطاف‌پذیری و هوشمندی کافی، در شرایط پویا عملکرد مطلوبی ندارند. برای رفع این محدودیت‌ها، مدل‌های ترکیبی ابر-اکتشافی و مبتنی بر قوانین ارائه شده‌اند که سعی در بهبود عملکرد این روش‌ها دارند. اگرچه این مدل‌ها تا حدی نتایج بهتری ارائه داده‌اند، اما همچنان در مواجهه با پیچیدگی‌های مسائل زمان‌بندی در محیط‌های متغیر و پویا، کارایی

[1] Cobots
[2] Flexible Job-Shop Scheduling Problem (FJSP)
[3] Multi-agent Reinforcement Learning (MARL)
[4] Makespan
[5] Rule-based Scheduling Methods (RBSM)
[6] Hyper-Heuristic Algorithms (HHA)

کافی ندارند [۸]. در سال‌های اخیر، یادگیری تقویتی[1] به‌عنوان یکی از روش‌های نوین برای حل مسائل پیچیده زمان‌بندی مطرح شده است. این روش با استفاده از تجربیات عامل‌ها، سیاست‌هایی نزدیک به بهینه را یاد می‌گیرد و در مسائل متنوعی مانند کنترل روبات‌ها، مدیریت وسایل نقلیه خودکار و زمان‌بندی تولید به‌کار گرفته شده است. ترکیب یادگیری تقویتی با یادگیری عمیق [۱۲]، پیشرفت‌های قابل‌توجهی را در حل مسائل بزرگ و پیچیده فراهم کرده است. برخی مطالعات نیز روش‌های یادگیری تقویتی را با قوانین اعزام وظایف و ابر-اکتشافی ترکیب کرده‌اند تا راه‌حل‌های بهتری ارائه دهند [۱۳]. سیستم‌های یادگیری تقویتی چندعاملی (MARL) نیز به دلیل توانایی آن‌ها در توزیع وظایف و خودمختاری عامل‌ها، به یکی از ابزارهای مهم در مدیریت محیط‌های صنعتی پویا تبدیل شده‌اند. این سیستم‌ها به عامل‌ها امکان می‌دهند که به‌صورت غیرمتمرکز عمل کنند و با همکاری یکدیگر به بهینه‌سازی زمان‌بندی بپردازند. مطالعات اخیر نشان داده‌اند که استفاده از مکانیزم‌های اختصاص اعتبار چندعاملی[2] (MCA) می‌تواند تعامل میان عامل‌ها و دینامیک محیط را بهبود بخشد. علاوه بر این، پژوهش‌های متعددی از چارچوب‌های مبتنی بر یادگیری تقویتی عمیق برای حل مسائل زمان‌بندی استفاده کرده‌اند [۱۴]. در این میان، ترکیب یادگیری تقویتی چندعاملی با محرک‌های انگیزه درونی، گامی نوین در بهبود عملکرد سیستم‌های زمان‌بندی محسوب می‌شود [۱۵]. این رویکرد نه تنها بهره‌وری سیستم را افزایش می‌دهد، بلکه به عامل‌ها امکان می‌دهد تا با استفاده از پاداش‌های درونی، وظایف خود را به‌صورت بهینه‌تری اولویت‌بندی کنند.

مطالعات نشان داده‌اند که استفاده از این رویکرد می‌تواند باعث بهبود چشم‌گیر در زمان‌بندی وظایف، کاهش زمان اجرای کل و افزایش بهره‌وری منابع شود [۱۶]. پژوهش ژانگ و همکاران [۱۷] مدلی مبتنی بر یادگیری تقویتی عمیق[3] و گراف‌های چندعاملی برای FJSP ارائه کرده است که در آن عامل‌ها وظایف را با توجه به توالی و مسیر بهینه انتخاب می‌کنند. مطالعات اخیر، مدل‌های مختلفی برای حل مسئله FJSP پیشنهاد کرده‌اند [۱۸]. یوهانسون و همکاران [۶] الگوریتمی مبتنی بر Double DQN برای زمان‌بندی پویا در یک سلول مونتاژ روباتیک توسعه داد. سان و همکاران [14] از یادگیری تقویتی چندعاملی برای تصمیم‌گیری در زمان واقعی در محیط‌های تولید با ظرفیت محدود استفاده کرد. کوین و همکاران [8] مدل DRL چندعاملی برای زمان‌بندی پویا ارائه کرد که به ترکیب آموزش استاتیک با اجرای پویا پرداخت. مدل پیشنهادی این پژوهش، با بهره‌گیری از محرک‌های انگیزه درونی و رویکرد CTDE، گامی نو در حل مسئله زمان‌بندی کارگاهی انعطاف‌پذیر محسوب می شود. این مدل با ترکیب سیاست‌های غیرمتمرکز و یادگیری عمیق، توانایی خود را در مدیریت وظایف پویا و بهینه‌سازی منابع به اثبات رسانده است.

۳. روش‌شناسی

۳،۱. بیان مسئله و ساختار سیستم

در این پژوهش، هدف اصلی ارائه یک مدل پویا برای مسئله زمان‌بندی کارگاهی انعطاف‌پذیر در یک محیط تولید چندعاملی است. این مدل، وظایف تولیدی مختلف را با استفاده از روبات‌های همکار (کوبوت‌ها) در ایستگاه‌های کاری پردازش می‌کند. در این محیط، وظایف تولیدی به‌صورت تصادفی و با نرخ ورود مشخصی (λ) بر اساس توزیع پواسون وارد سیستم می شوند [۱۹]. وظایف به صورت مجموعه‌ای از عملیات تعریف می شوند که باید در ایستگاه‌های کاری مشخص و با استفاده از منابع موجود انجام شوند. هر وظیفه $mj_i \epsilon MJ$ دارای یک تاریخ سررسید $T_{mj_i}^{DD}$ است که باید

[1] Reinforcement Learning (RL)

[۲] Multi-agent Credit Assignment (MCA)

[3] Deep Reinforcement Learning (DRL)

تا آن زمان تکمیل شود. وظایف بین ایستگاه‌های کاری توسط کوبوت‌ها منتقل می‌شوند، و هر کوبوت وظایف خاصی را مدیریت می‌کند. نقش کلیدی کوبوت‌ها، کاهش تأخیرها و بهینه‌سازی فرآیند انتقال وظایف بین ایستگاه‌ها است.

۳،۲. محدودیت‌ها و مفروضات مدل

برای طراحی یک سیستم مؤثر، مفروضات در نظر گرفته شده شامل: i) پردازش وظایف: هر وظیفه از مجموعه‌ای از عملیات تشکیل شده است که باید به‌ترتیب مشخصی پردازش شوند، ii) قابلیت کوبوت‌ها: چندین کوبوت مشابه در سیستم وجود دارد، اما هر کوبوت تنها می‌تواند یک عملیات را در هر لحظه انجام دهد، iii) ظرفیت ایستگاه‌های کاری: هر ایستگاه کاری فقط یک عملیات را در هر لحظه پردازش می‌کند، iv) محدودیت منابع: منابع موجود در هر ایستگاه کاری ممکن است در هر زمان خاص محدود باشند و v) زمان لجستیک: زمان انتقال وظایف بین ایستگاه‌های کاری متناسب با فاصله آن‌ها تعیین می‌شود، اگرچه عوامل دیگری نیز ممکن است این زمان را تحت تأثیر قرار دهند. توضیحات نمادها در جدول ۱ ارائه شده‌اند.

۳،۳. تابع هدف: کاهش زمان کل اجرای وظایف

تابع هدف اصلی، حداقل سازی زمان کل مورد نیاز برای تکمیل تمامی وظایف است [۲۰]. این زمان شامل سه بخش است: i) زمان انتقال وظایف توسط کوبوت‌ها، ii) زمان توقف عملیات در صورت بروز اختلالات و iii) زمان پردازش عملیات در ایستگاه‌های کاری. فرض کنیم $\mathrm{mw}_{i,k}$ و $\mathrm{mr}_{i,l}$ ایستگاه کاری و منبع انتخابی برای وظیفه i در مرحله t باشند. بنابراین، تابع زمان کل اجرای وظایف به صورت فرمول ۱ تعریف می‌شود که مجموع زمان مصرف‌شده برای هر عملیات در وظایف تولیدی است:

$$\min Obj_{makespan} = \max_{mj \in MJ} \left(\sum_{t \in T} T^{lgs}_{mw_{t,i}} + T^{P}_{mw_{t,k}, mr_{t,l}} + T^{sus}_{moj_t} \right) \quad (1)$$

با این شرط که:

$$\forall mj_i \in MJ, \quad \sum_{moj \in mj_i} T^{P}_{moj} < T^{DD}_{mj_i}$$

جدول ۱. نمادهای به‌کاررفته در توصیف محیط مسئله.

شاخص‌ها	ثابت‌ها
شاخص وظایف : $\boldsymbol{i}$	تعداد وظایف تولیدی : $\boldsymbol{n^{mj}}$
شاخص عملیات : $\boldsymbol{j}$	تعداد ایستگاه‌های کاری : $\boldsymbol{n^{mw}}$
شاخص کوبوت : $\boldsymbol{q}$	تعداد کوبوت ها : $\boldsymbol{n^{c}}$
شاخص ایستگاه‌های کاری : $\boldsymbol{k}$	نرخ ورود وظایف : $\boldsymbol{\lambda}$
شاخص منابع : $\boldsymbol{l}$	
مجموعه‌ها	
مجموعه وظایف تولیدی : $\boldsymbol{MJ}$	
$MJ = \{mj_1, mj_2, .., mj_{n^{mj}}\}$	

مجموعه ایستگاه‌های کاری : $\boldsymbol{MW}$

$$MW = \{mw_1, mw_2, .., mw_{n^{mw}}\}$$

مجموعه‌کوبوت ها : $\boldsymbol{C}$

$$\boldsymbol{C} = \{\boldsymbol{C_1}, \boldsymbol{C_2}, .., \boldsymbol{C_{n^c}}\}$$

مجموعه عملیات : $\boldsymbol{MOJ}$

$$MOJ = \{moj_j | j = 1,2, ..\}$$

توزیع منابع در ایستگاهای کاری : $\boldsymbol{\mathcal{D}_R}$

ماتریس زمان انتقال کوبوت ها : $\boldsymbol{lgs^c}$

توالی عملیات وظیفه $\mathbf{i}$ م : $\boldsymbol{Seq_i}$

مجموعه منابع : $\boldsymbol{MR}$

$$.MR = \{mr_l | l = 1,2, ..\}$$

متغیرها

انتقال وظایف توسط کوبوت $\boldsymbol{q}$بین ایستگاه کاری $\boldsymbol{k}$ و $\boldsymbol{k'}$ $\boldsymbol{lgs^{c_q}_{mw_k, mw_{k'}}}$

عملیات $\mathbf{j}$از وظیفه $\mathbf{i}$: : $\boldsymbol{moj_{i,j}}$

زمان لجستیک ایستگاه کاری $\boldsymbol{l}$ در زمان $\mathbf{t}$: $\boldsymbol{T^{lgs}_{mw_{t,i}}}$

زمان پردازش ایستگاه کاریk در زمان $\mathbf{t}$: $\boldsymbol{T^{P}_{mw_{t,k}, mr_{t,l}}}$

زمان کل تعلیق عملیات در زمان $\mathbf{t}$: $\boldsymbol{T^{sus}_{moj_t}}$

تاریخ سر رسید وظیفه $\mathbf{i}$: $\boldsymbol{T^{DD}_{mj_i}}$

۳،۴. معماری سیستم یادگیری تقویتی چندعاملی

سیستم یادگیری تقویتی چندعاملی پیشنهادی در این پژوهش، بر همکاری عامل‌ها برای حل مسئله زمان‌بندی متمرکز است. در این معماری، هر وظیفه به‌عنوان یک عامل مستقل در نظر گرفته می شود که به‌طور غیرمتمرکز عمل کرده و تصمیم‌گیری‌های خود را بر اساس وضعیت محیط و سیاست‌های یادگیری اتخاذ می‌کند [۲۱]. این سیستم، با ترکیب مؤلفه‌های یادگیری انگیزه درونی و یادگیری تقویتی، بهینه‌سازی فرآیند زمان‌بندی را امکان‌پذیر می‌کند. معماری سیستم شامل سه بخش اصلی است که در تعامل با یکدیگر، عملکرد کلی سیستم را بهبود می‌بخشند. نخستین بخش، سیاست یادگیری غیرمتمرکز است که به هر عامل اجازه می‌دهد به‌طور مستقل از تجربیات خود برای یادگیری و بهبود تصمیم‌گیری استفاده کند. عامل‌ها بدون وابستگی به یک کنترل‌کننده مرکزی، از اطلاعات محلی مانند وضعیت ایستگاه‌های کاری، پیشرفت وظایف، و اولویت‌بندی عملیات بهره می‌برند تا بهترین اقدام ممکن را انتخاب کنند. بخش دوم، محرک انگیزه درونی است که برای هر عامل، یک مکانیزم پاداش درونی فراهم می‌کند. این مکانیزم، عامل‌ها را تشویق می‌کند تا با اولویت‌بندی بهینه وظایف، از تأخیرهای احتمالی جلوگیری کنند. پاداش درونی مبتنی بر وضعیت و اقدام عامل‌ها تعریف می‌شود و مکمل پاداش‌های بیرونی محیط است. این ویژگی به عامل‌ها کمک می‌کند تا در کنار بهبود عملکرد فردی، هماهنگی بهتری با سایر عامل‌ها داشته باشند. در نهایت، بخش سوم شامل منتقد متمرکز است که به‌عنوان یک ناظر عمل کرده و عملکرد عامل‌ها را ارزیابی می‌کند. منتقد متمرکز، با استفاده از اطلاعات جمع‌آوری‌شده از محیط و تعاملات عامل‌ها، سیاست‌های یادگیری را به‌روزرسانی می‌کند. این به‌روزرسانی‌ها به عامل‌ها کمک می‌کند تا رفتار خود را در جهت بهبود عملکرد کلی سیستم تنظیم کنند. به‌طور کلی، این معماری با ترکیب یادگیری غیرمتمرکز، انگیزه درونی، و بازخورد منتقد متمرکز، یک رویکرد نوآورانه برای بهینه‌سازی زمان‌بندی در

محیط‌های تولید پویا ارائه می‌دهد. عامل‌ها علاوه بر تمرکز بر وظایف فردی خود، با سایر عامل‌ها نیز تعامل کرده و در جهت کاهش زمان کل اجرای وظایف و بهبود بهره‌وری منابع تلاش می‌کنند.

۳.۵. فرآیند تصمیم‌گیری مارکوف

به‌جای بهینه‌سازی مستقیم هدف از طریق مدل‌های ریاضی یا الگوریتم‌های ابتکاری، MARL مسائل تصمیم‌گیری متوالی را با مدل‌سازی آن‌ها به‌عنوان یک MDP حل می‌کند[۱۳] . در این پژوهش، ما از یک سیستم کاملاً مشارکتی چندعاملی استفاده می‌کنیم که از الگوریتم DRL غیرمتمرکز بهره می‌برد. در این سیستم، عامل‌ها باید به‌صورت مستقل عمل کنند و بدون کنترل مرکزی، سیاست‌های بهینه تصمیم‌گیری را برای هر وظیفه یاد بگیرند. مدل پیشنهادی بر اساس فرآیند تصمیم‌گیری مارکوف غیرمتمرکز (Dec-POMDP) طراحی شده است. یک وظیفه تصمیم‌گیری در این زمینه به‌صورت یک Dec-POMDP تعریف می‌شود که شامل مجموعه زیر است:

$< \mathcal{S}, \mathcal{A}, \mathcal{U}, \mathcal{T}, O, P, r^{ext}, \Upsilon >$ که $\mathcal{S}$ مجموعه‌ای از مشاهدات جزئی محیط توسط عامل‌ها، $\mathcal{A}$ مجموعه‌ای از عامل‌ها، $\mathcal{U}$ مجموعه اقدامات ممکن، $\mathcal{T}$ تابع انتقال وضعیت، r^{ext} تابع پاداش بیرونی و Υ عامل تخفیف برای پاداش‌های آینده است. این مدل، O به‌عنوان تابع مشاهده تعریف می‌شود که اطلاعات جزئی هر عامل را از وضعیت کلی محیط استخراج می‌کند، مانند وضعیت ایستگاه کاری مقصد، تعداد وظایف در صف، و زمان‌بندی‌های قبلی. پارامتر P نیز نشان‌دهنده احتمال مشاهده وضعیت t^S توسط عامل i با استفاده از مشاهده محلی o_t^i است. این تابع نقش کلیدی در تعریف عدم قطعیت مشاهده‌ای عامل‌ها ایفا می‌کند، به‌ویژه در محیط‌هایی با دید ناقص .در هر لحظه، عامل‌ها مجموعه مشاهدات محلی خود را بررسی کرده و اقداماتی انجام می‌دهند که منجر به تغییر وضعیت محیط می‌شود. هدف هر عامل، حداکثرسازی مجموع پاداش‌های آینده است.

هدف هر عامل، حداکثرسازی مجموع پاداش‌های آینده است.

۳.۶. تابع پاداش

تابع پاداش بررسی می‌کند که آیا تصمیمات گرفته‌شده توسط شبکه صحیح هستند یا خیر. برای ارزیابی عملکرد عامل‌ها، تابع پاداش زیر تعریف شده است: i) پاداش مثبت برای تکمیل وظایف بدون تأخیر، ii) پاداش منفی برای وظایف به تأخیر افتاده یا متوقف شده. تابع پاداش به‌صورت فرمول ۲ بیان می‌شود:

$$r_t = \begin{cases} N_t^{fin} + N_t^{halt} - N_t^{DD} - \dfrac{t}{k_1}, & t \neq t_{done} \\ \dfrac{k_2}{T^{total}}, & t = t_{done} \end{cases} \quad (2)$$

که در آن دراینجا N_t^{fin} نشان‌دهنده تعداد وظایفی است که در گام زمانی t تکمیل شده‌اند. این مقدار به‌صورت جمع تمامی اقداماتی است که $u_t = act_t(finish)$ باشند و $u_t \in U$، از طرفی N_t^{halt} تعداد وظایفی است که در گام زمانی t متوقف شده‌اند و این مقدار نیز به‌صورت جمع تمامی اقداماتی است که $u_t = act_t(halt)$ باشند و $u_t \in U$، همچنین N_t^{DD} تعداد وظایفی است که از تاریخ سررسید خود فراتر رفته و تکمیل نشده‌اند .علاوه بر این k_1, k_2 پارامترهای مثبت صحیح ($\in \mathbb{Z}^+$) هستند که وزن اهمیت دو مؤلفه پاداش (پاداش مثبت برای اتمام موفق وظایف و پاداش نهایی پایان اپیزود) را تعیین می‌کنند. مقداردهی این ضرایب به‌صورت تجربی و با توجه به هدف کنترل شدت اثرگذاری هر بخش انجام شده است. بر اساس تحلیل حساسیت، مقادیر 1 و 5 به ترتیب برای k_1 و k_2 پایداری بهتری در فرآیند آموزش ایجاد کردند. T^{total} نشان‌دهنده زمان کل اجرای تمامی وظایف است، یعنی مجموع

زمان‌های لازم برای تکمیل تمام عملیات در سیستم. این مقادیر برای ارزیابی عملکرد سیستم در هر گام زمانی و بهینه‌سازی فرآیند زمان‌بندی استفاده می‌شوند.

۳,۷. بازنویسی معماری الگوریتم با ساختار جدید

در این پژوهش، برای حل مسئله زمان‌بندی کارگاهی انعطاف‌پذیر در محیط‌های تولید پویا، یک معماری یادگیری تقویتی چندعاملی معرفی شده است که از مدل یادگیری پاداش درونی فردی (LIIR) الهام گرفته شده است [٤]. این رویکرد ترکیبی از یادگیری غیرمتمرکز، پاداش‌های انگیزه درونی و سیستم‌های منتقد بیرونی را برای بهینه‌سازی تصمیم‌گیری عامل‌ها و بهبود کارایی سیستم ارائه می‌دهد. مزیت اصلی این معماری، توانایی آن در مدل سازی رفتار عامل‌ها در محیط‌های پویا و پیچیده با کمترین نیاز به ساده‌سازی‌های مصنوعی است. معماری پیشنهادی با تمرکز بر سه اصل طراحی شده است: i) عامل‌ها بدون وابستگی به یک کنترل‌کننده مرکزی، بر اساس سیاست‌های مستقل تصمیم‌گیری می‌کنند. این سیاست‌ها از طریق تعاملات محلی با محیط و سایر عامل‌ها بهبود می‌یابد، ii) استفاده از پاداش‌های انگیزه درونی به عامل‌ها کمک می‌کند تا علاوه بر تمرکز بر اهداف کوتاه‌مدت، بر اهداف بلندمدت سیستم نیز تمرکز داشته باشند، iii) ارزیابی و بهینه‌سازی دو سطحی به دین معنی که فرآیند یادگیری شامل دو سطح است: حداکثرسازی پاداش‌های فردی عامل‌ها و بهینه‌سازی پاداش‌های کلی سیستم. در این معماری، پاداش کلی برای عامل‌ها به دو بخش اصلی تقسیم می شود: پاداش بیرونی و پاداش درونی. پاداش بیرونی (r_t^{ext}) از محیط دریافت می شود و مرتبط با عملکرد کلی سیستم است. پاداش درونی ($r_{\omega_{i,t}}^{in}$) بر اساس اولویت‌های عامل و شرایط محلی تعریف می‌شود. برای ترکیب این دو نوع پاداش، یک تابع پاداش پروکسی ($r_{i,t}^{proxy}$) تعریف شده است که به صورت زیر در فرمول ۳ بیان می‌شود:

$$r_{i,t}^{proxy} = r_t^{ext}(\alpha - 1) + r_{\omega_{i,t}}^{in}(s_{i,t}, u_{i,t}) * \alpha \qquad (3)$$

در این فرمول، α یک ابرپارامتر برای تنظیم تأثیر پاداش درونی نسبت به پاداش بیرونی است. این رویکرد به عامل‌ها اجازه می‌دهد که ضمن بهینه سازی اهداف کوتاه‌مدت خود، به اهداف بلندمدت سیستم نیز توجه کنند. مقدار پارامتر α که وزن ترکیب پاداش درونی و بیرونی را تعیین می‌کند، در این پژوهش به صورت تجربی تنظیم شده است. برای تعیین مقدار بهینه، تحلیل حساسیت در بازه [۰,۲ ,۰,۸] با گام ۰,۱ انجام شد و نتایج نشان دادند که بازه ۰,۴ تا ۰,۶ عملکرد پایدارتری برای مدل ایجاد می‌کند. در نهایت، مقدار $\alpha = 0.5$ انتخاب و برای تمامی آزمایش‌ها استفاده گردید.

۳,۸. ساختار سیستم یادگیری تقویتی

معماری سیستم شامل سه بخش اصلی است، i) سیاست عامل‌ها و یادگیری مستقل: هر عامل در سیستم، به‌عنوان یک شبکه عصبی سه‌لایه مدل سازی می شود که شامل لایه‌های ورودی، بازگشتی و خروجی است. داده‌های ورودی شامل مشاهدات محیطی، کدگذاری تک‌حالته نمایه عامل و اقدام قبلی آن است. این اطلاعات پس از پردازش در لایه بازگشتی دروازه‌دار(GRU) ، به‌عنوان یک تابع سیاست به عامل کمک می‌کند که اقدام بهینه را انتخاب کند. لایه خروجی شبکه با استفاده از یک تابع فعال سازی Leaky ReLU طراحی شده است تا از مشکلات مرتبط با انفجار گرادیان جلوگیری کند، ii) پاداش درونی و نقش آن در یادگیری: پاداش درونی یکی از اجزای کلیدی این معماری است که برای هر عامل به‌صورت مستقل تعریف می‌شود. این پاداش‌ها به عامل‌ها کمک می‌کنند تا با اولویت‌بندی بهتر وظایف خود، تأخیرها را کاهش دهند. از آنجا که تعریف دقیق پاداش درونی ممکن است چالش‌برانگیز باشد، یک تابع پاداش پروکسی طراحی شده است که پیچیدگی محیط را بدون نیاز به تعریف جفت حالت-اقدام منعکس می‌کند. این

تابع، پاداش درونی و بیرونی را ترکیب کرده و به عامل‌ها انگیزه می‌دهد تا رفتار بهینه‌ای داشته باشند،iii) منتقد بیرونی و ارزیابی سیستم: برای ارزیابی عملکرد عامل‌ها، یک شبکه منتقد بیرونی طراحی شده است که به تخمین ارزش حالت‌ها و اقدامات در محیط می‌پردازد. این شبکه از تابع مزیت (A_π) استفاده می‌کند که به هر عامل اجازه می‌دهد عملکرد خود را در مقایسه با اقدامات میانگین ارزیابی کند. این تابع به صورت فرمول ۴ تعریف می‌شود:

$$A_\pi(s,u) = r^{ex}(s,u) + V_\partial^{ex}(s') - V_\partial^{ex}(s) \qquad (4)$$

در این فرمول، s حالت فعلی، s' حالت بعدی پس از انجام اقدام u ، و V_∂^{ex} ارزش حالت بر اساس شبکه منتقد بیرونی ا ست. شبکه منتقد بیرونی، ارزش حالت‌ها را تخمین زده و با ا ستفاده از تابع مزیت، اقدامات عامل‌ها را ارزیابی می‌کند. این ارزیابی به عامل‌ها کمک می‌کند که سیاست‌های خود را بهینه کنند. علاوه بر این، یک شبکه منتقد پروکسی نیز طراحی شده ا ست که ارزش حالت‌ها را از دیدگاه عامل محا سبه می‌کند. تابع مزیت پروکسی به صورت فرمول شماره ۵ تعریف می‌شود:

$$A_i^{proxy}(s_i,u_i) = r_i^{proxy}(s_i,u_i) + V_{\vartheta_i}^{proxy}(s') - V_{\vartheta_i}^{proxy}(s) \qquad (5)$$

در اینجا، $V_{\vartheta_i}^{proxy}$ نشان‌دهنده پارامتر ارزش پروکسی برای عامل i است و s' حالت بعدی عامل i در مسیر حرکت آن را ن شان می‌دهد. ما مدل سیا ست و سی ستم MASS را به‌طور همزمان با ا ستفاده از پس‌انت شار گرادیان و نزول گرادیان در الگورتیم ۱ آموزش می‌دهیم. برای هر عامل، پاداش درونی به صورت م ستقل تعریف شده و با هدف هدایت رفتار عامل در جهت کاهش تأخیر، مدیریت بار ای ستگاه کاری و رعایت اولویت وظایف طراحی شده ا ست. پارامترهایی مانند میزان تأخیر وظیفه، تراکم ایستگاه مقصد، اهمیت نسبی وظیفه و نرخ ورود فعلی وظایف در تعریف این پاداش مؤثر هستند. این مقادیر ابتدا به‌عنوان ورودی به یک تابع غیرخطی مبتنی بر شبکه عصبی سبک وارد می‌شوند و خروجی آن به‌عنوان پاداش درونی مورد استفاده قرار می‌گیرد. این پاداش سپس با پاداش بیرونی محیط ترکیب شده و تابع نهایی پاداش پروکسی r_i^{proxy} را شکل می‌دهد، که مبنای به‌روز سانی سیا ست‌های عامل‌ها ا ست. لازم به ذکر ا ست که در این مدل، شبکه منتقد بیرونی صرفاً در مرحله آموزش به‌کار گرفته می‌شود و در زمان اجرا، تمامی عامل‌ها به صورت م ستقل و غیرمتمرکز، بر ا ساس م شاهدات محلی خود اقدام می‌کنند. این طراحی منطبق با رویکرد CTDE بوده و از نظر عملیاتی با الزامات محیط‌های واقعی صنعتی سازگار است.

۳،۹. فرآیند آموزش معماری

الگوریتم آموزشی پیشنهادی به‌صورت کلی به سه بخش تقسیم می شود:i) مقداردهی اولیه :پارامترهای شبکه‌های سیاست، پاداش درونی و حافظه بازپخش مقداردهی می‌شوند، ii) تعامل عامل‌ها با محیط :عامل‌ها بر اساس سیا ست‌های فعلی خود، اقداماتی انجام می‌دهند که منجر به تغییر و ضعیت محیط و ثبت تجربیات جدید می شود،iii) به‌روز سانی سیا ست‌ها :با ا ستفاده از داده‌های ثبت شده در حافظه بازپخش، پارامترهای شبکه‌های سیا ست و پاداش درونی به‌روزرسانی می‌شوند. این به‌روزرسانی‌ها بر اساس الگوریتم نزول گرادیان و توابع مزیت انجام می‌شود.

۳،۱۰. مزایای معماری پیشنهادی

این معماری قادر است در محیط‌های تولید پویا و با تعداد زیادی از عامل‌ها عملکرد مطلوبی داشته باشد.با استفاده از ترکیب پاداش‌های درونی و بیرونی، عامل‌ها به‌طور همزمان بر اهداف فردی و سیستم تمرکز دارند و استفاده از شبکه‌های بازگشتی و توابع مزیت به بهینه سازی فرآیند تصمیم‌گیری کمک می‌کند. معماری پیشنهادی ن شان‌دهنده یک گام رو به جلو در استفاده از یادگیری تقویتی برای حل مسائل پیچیده زمان‌بندی در محیط‌های صنعتی است. این روش نه تنها عملکرد فردی عامل‌ها را بهبود می‌بخشد، بلکه بهره‌وری کلی سیستم را نیز ارتقا می‌دهد.

۴. مطالعه تجربی: ارزیابی عملکرد مدلMASS

برای ارزیابی مدل پیشنهادیMASS، سناریوهای مختلفی از مسئله زمان‌بندی کارگاهی انعطاف‌پذیر شبیه سازی شده ا ست. در این مطالعه، شاخص ا صلی برای اندازه‌گیری عملکرد، زمان کل اجرای وظایف ا ست. ما عملکرد روش MASSرا با الگوریتم‌های شناخته‌شده‌ای مانندPDR ، HHAو MADRL مقایسه کرده و آزمایش‌های مختلفی برای بررســـی پایداری آموزش، عملکرد، و مقیاس‌پذیری آن انجام داده‌ایم. نتایج زمان‌بندی از طریق نمودارهای گانت برای نمایش دقیق ارائه شده‌اند.

۴،۱. پلتفرم توسعه

شبیه‌سازی‌ها بر روی پلتفرمی مبتنی بر Python 2.7.3 و PyTorch 1.4.1 اجرا شده‌اند. سخت‌افزار استفاده‌شده شامل یک کارت گرافیک NVIDIA GeForce RTX 2080Ti با ۱۶ گیگابایت حافظه و یک پردازنده Intel Core i7-9700KF ست. محیط آزمایش با ا ستفاده از بسته OpenAI Gym طراحی شده که ابزارهای لازم برای تو سعه محیط‌های یادگیری تقویتی را فراهم می‌کند. زمان مصرف‌شده در این آزمایش‌ها به دقیقه محاسبه شده است.

الگوریتم ۱: الگوریتم آموزشMASS

۱. تا زمانی که شرط خاتمه برقرار نشده است، انجام دهید:
   ۱. بازنشانی محیط .MASS
   ۲. مقداردهی اولیه:
      - پارامترهای سیاست بازیگرهای زمان‌بندی θ
      - پارامتر پاداش درونی ω
      - حافظه بازپخش $\mathcal{D}$
   ۲. برای هر t=1 تا max_episode، انجام دهید.
   ۱. مشاهدات عامل o_i از محیط را دریافت کنید.
   ۲. برای هر بازیگر (actor) در گام زمانی t
      - از شبکه سیاست برای انتخاب اقدامات موجود $u_i|o_o$ استفاده کنید.
      - اقدام u_i را در محیط اجرا کنید.
      - وضعیت جدید o_i' و مشاهدات فعلی r_i^{ex} را مشاهده کنید.
   ۳. پاداش درونی r_i^{in} را محاسبه کنید.
   ۴. تجربه $(o_i , r_i^{in} , r_i^{ext} , o_i' , s_{tr})$ را در حافظه $\mathcal{D}$ ذخیره کنید.
   ۳. نمونه‌برداری از یک مینی‌بچ (Minibatch) از حافظه $\mathcal{D}$:
   ۱. به‌روزرسانی ω با استفاده از تابع مزیت و نرخ یادگیری β طبق رابطه :(8)

$$\omega' = \omega + \beta \nabla \log(\pi|s) A_\pi(s,u)$$

   ۲. به‌روزرسانی θ با استفاده از تابع مزیت پروکسی و نرخ یادگیری α طبق رابطه : (9)

$$\theta' = \theta + \alpha \nabla \log(\pi|s) A^{proxy}(s,u)$$

پایان الگوریتم.

۴،۲. داده‌های شبیه‌سازی‌شده

وظایف تولیدی در طول شبیه سازی‌ها بر ا ساس توزیع پوا سون با نرخ ورود λ وارد سی ستم می شوند. برای هر وظیفه، احتمال انتخاب یک سانی برای مح صولات مختلف وجود دارد. فرض شده ا ست که تعداد کوبوت‌ها برابر با تعداد ای ستگاه‌های کاری ا ست و زمان لج ستیک در بازه‌ای از ۱ تا ۷ قرار دارد. داده‌های شبیه سازی شده به صورت ت صادفی تولید شده و رویدادهای احتمالی بر ا ساس فرآیند پوا سون تعریف شده‌اند. نرخ ورود λ_t در هر زمان t بر ا ساس فرمول شماره ۶ محاسبه می‌شود:

$$\lambda_t = \frac{\sum_{i=1}^{n^{mj}} \sum_{j=1}^{n^{moj}} T_{i,j}^{P} + \sum_{k=1}^{n^{mw}} \sum_{k\prime=1}^{n^{mw}} T_{mw_{t-1,k},mw_{t-1,k\prime}}^{lgs}}{n^{mj} n^{mw} \Omega + \mathrm{n}^{c} \upsilon} \qquad (6)$$

در اینجا، Ω نرخ ا ستفاده از ای ستگاه کاری، υ نرخ ا ستفاده از کوبوت‌ها ا ست و به صورت ثابت برابر با ۹۵٪ در نظر گرفته شده است. توضیحات سایر نمادها در جدول ۱ ارائه شده‌اند. لازم به ذکر است که در این شبیه‌سازی‌ها، تنها ورود وظایف به‌صورت تصادفی و پویا در نظر گرفته شده و اختلالاتی نظیر خرابی منابع یا توقف ناگهانی ایستگاه‌ها مدل‌سازی نشـده‌اند. هدف اصـلی در این مرحله، ارزیابی عملکرد مدل در برابر نوسـانات ورودی و بررسـی پایداری آموزش عامل‌ها بوده است. با این حال، معماری پیشنهادی قابلیت گسترش به سناریوهای شامل اختلالات عملیاتی را نیز دارا می‌باشد. انتخاب توزیع پوا سون برای مدل سازی ورود وظایف، مبتنی بر ویژگی‌های آماری شناخته شده این توزیع در مدل‌سازی ورود گسسته سیستم‌های تولیدی است. این انتخاب با مطالعات پیشین در حوزه زمان‌بندی تولید هم‌راستا بوده و امکان تنظیم دقیق نرخ ورود و بررسی پایداری عملکرد سیستم در شرایط مختلف را فراهم کرده است.

۴،۳. محدودیت زمان سررسید(DDT)

در شبیه‌سازی‌ها، محدودیت زمان سررسید (DDT)[1] به‌عنوان میزان زمانی تعریف می‌شود که وظیفه می‌تواند قبل از رسیدن به تاریخ سررسید تکمیل شود. تاریخ سررسید $T_{mj_i}^{DD}$ هر وظیفه با استفاده از فرمول شماره ۷ تعیین می‌شود:

$$T_{mj_i}^{DD} = A_{mj_i} + \left(\sum_{j=1}^{moj_i} T_{mj_i,j}^{P} \,.\, \mathrm{DDT} \right) \qquad (7)$$

که در آن: A_{mj_i} زمان ورود وظیفه، $T_{mj_i,j}^{P}$ زمان پردازش عملیات، DDT پارامتری که نشـان‌دهنده سـختی یا انعطاف‌پذیری زمان سررسید است. نتایج شبیه‌سازی برای مقادیر DDT برابر با ۱، ۱،۵ و ۲ مورد بررسی قرار گرفته‌اند.

۴،۴. نمونه‌های مسئله

جدول ۲ تنظیمات اصـلی نمونه‌های شـبیه‌سـازی‌شـده را نشـان می‌دهد. این نمونه‌ها شـامل تعداد وظایف، تعداد ایسـتگاه‌های کاری، و دیگر ویژگی‌های مرتبط با عملیات هسـتند. لازم به ذکر اسـت که با وجود انعطاف‌پذیری مدل MASSدر برابر تغییر مقیاس سـیسـتم، برای انطباق کامل با پیکربندی جدید ایسـتگاه‌های کاری یا کوبوت‌ها، نیاز به آموزش مجدد یا تنظیم مجدد وزن‌های شبکه‌ها وجود دارد. این ویژگی در چارچوب مدل پیش‌بینی شده و با شرایط جدید سازگار می گردد.

۴،۵. الگوریتم‌های مقایسه‌ای، آموزش و آزمون

برای برر سی اثربخ شی و توانایی مدل MASS در حل م سائل زمان‌بندی کارگاهی انعطاف‌پذیر، مقای سه‌ای جامع میان این مدل و چند الگوریتم معتبر دیگر انجام شده است. این الگوریتم‌ها به دلیل کاربرد گسترده و نتایج برجسته در

[1] Due Date Tightness (DDT)

مسائل مشابه انتخاب شده‌اند. یکی از این الگوریتم‌ها، QMIX است که به دلیل قابلیت‌های بالا در مدیریت سیستم‌های چندعاملی کوچک و استفاده از یک شبکه ترکیبی برای تصمیم‌گیری متمرکز شناخته شده است. این الگوریتم در شرایطی که تعداد عامل‌ها محدود بوده و پیچیدگی محیط کمتر است، عملکرد بسیار خوبی از خود نشان می‌دهد. در مقابل، مدل MASS برای محیط‌های یادگیری تقویتی چندعاملی غیرمتمرکز طراحی شده است که در آن، هر عامل به‌صورت مستقل عمل کرده و از تعامل با سایر عامل‌ها یاد می‌گیرد.علاوه برQMIX ، یک روش ابر-اکتشافی مبتنی بر بازپخت شبیه‌سازی‌شده (SA-HH) نیز برای مقایسه انتخاب شده است. این روش به دلیل توانایی در جستجوی فضای راه‌حل و یافتن نتایج نزدیک به بهینه در مسائل پیچیده زمان‌بندی مورد توجه قرار گرفته است. همچنین، دو روش مبتنی بر قوانین اولویت‌بندی (PDR) شامل SPT (کوتاه‌ترین زمان پردازش) و EDD (اولین تاریخ سررسید) به‌عنوان الگوریتم‌های پایه‌ای و پرکاربرد در محیط‌های تولید واقعی در این مقایسه گنجانده شده‌اند. این دو روش به‌طور گسترده در صنعت استفاده می‌شوند و به دلیل سادگی و سرعت اجرا، در شرایطی که نیاز به تصمیم‌گیری سریع است، کاربرد دارند. برای بررسی قابلیت‌ها و پایداری الگوریتم MASS در مقایسه باQMIX ، از فرآیندهای آموزشی استاندارد استفاده شد. یکی از معیارهای مهم در این آزمایش‌ها، پایداری آموزش است.

جدول ۲. مشخصات ورودی نمونه‌های زمان‌بندی برای شبیه‌سازی.

نمونه‌ها	تعداد وظایف (n^{mj})	تعداد ایستگاه کاری (n^{mw})	تعداد عملیات (n^{moj})	حداکثر منابع ($\max(n^{MR})$)	زمان پردازش ($T^{P}_{(.)}$)
D01	۱۰	۶	بازه تصادفی[5-7]	۳	بازه تصادفی[1-3]
D02	۱۰	۶	بازه تصادفی [5-7]	۶	بازه تصادفی[1-3]
D03	۱۵	۸	ثابت[10]	۵	بازه تصادفی[1-3]
D04	۱۵	۸	بازه تصادفی[3-10]	۳	بازه تصادفی[1-3]
D05	۱۵	۴	بازه تصادفی[5-10]	۲	بازه تصادفی[3-5]
D06	۱۰	۱۵	ثابت[15]	۵	بازه تصادفی[1-3]
D07	۲۰	۵	بازه تصادفی[5-10]	۵	بازه تصادفی[1-3]
D08	۲۰	۱۰	بازه تصادفی[5-10]	۲	بازه تصادفی[1-3]
D09	۲۰	۱۰	ثابت[10]	۵	بازه تصادفی[1-3]
D10	۲۰	۱۵	ثابت[10]	۲	بازه تصادفی[1-3]
D11	۳	۲	بازه تصادفی [5-7]	۲	بازه تصادفی[1-3]

پایداری در آموزش نه‌تنها به‌عنوان شاخصی از تکرارپذیری مدل بلکه به‌عنوان معیاری برای ارزیابی قابلیت اعتماد به آن نیز اهمیت دارد MASS .به دلیل استفاده از یک چارچوب غیرمتمرکز، توانایی بیشتری در مدیریت محیط‌هایی با تعداد زیاد عامل‌ها و وظایف پیچیده دارد. این امر ناشی از طراحی خاص مدل برای سازگاری با محیط‌های پویا و غیرمتمرکز است که در آن عامل‌ها از مشاهدات محلی خود یاد می‌گیرند. برای ارزیابی دقیق عملکرد، فرآیندهای آموزشی و آزمون در محیط‌های شبیه‌سازی‌شده انجام شد. محیط‌های آزمایشی انتخاب‌شده شامل سه نمونه شامل D01، D02، و D06 از جدول داده‌های شبیه‌سازی‌شده بودند. این نمونه‌ها به‌گونه‌ای انتخاب شده‌اند که تنوع در تعداد عامل‌ها، ایستگاه‌های کاری، و شرایط منابع را پوشش دهند. هدف این بود که تأثیر تغییرات در تعداد ایستگاه‌های کاری و منابع را بر عملکرد الگوریتم‌ها ارزیابی کنیم. برای هر الگوریتم، پاداش اپیزودیک (مجموع پاداش‌های دریافتی در یک اپیزود) به‌عنوان معیار اصلی عملکرد در نظر گرفته شد. پاداش اپیزودیک بالاتر نشان‌دهنده عملکرد بهتر مدل است.

نتایج نشـان داد که در محیط‌هایی با تعداد محدود منابع یا ایسـتگاه‌های کاری، عملکرد MASS و QMIX در مراحل اولیه آموزش مشابه است. بااین‌حال، در محیط‌های پیچیده‌تر و با تعداد عامل‌های بیشتر، MASS به‌طور قابل‌توجهی QMIX را پشت سر گذاشت. این عملکرد بهتر ناشی از مکانیزم انگیزه درونی و معماری یادگیری پیشرفته MASS اســت که به عامل‌ها کمک می‌کند تا رفتارهای خود را بهینه کرده و از تأخیر جلوگیری کنند. به‌طورکلی، مقایســه‌ها نشان می‌دهد که MASS علاوه بر ارائه نتایج پایدارتر، توانایی مقیاس‌پذیری بهتری نسبت به QMIX و سایر روش‌ها دارد. این یافته‌ها نشان‌دهنده پتانسیل بالای مدل پیشنهادی برای مدیریت مسائل پیچیده زمان‌بندی در محیط‌های تولیدی مدرن است. جدول ۳ مقایسه‌ای از عملکرد روش‌های مختلف زمان‌بندی شامل PDR، الگوریتم ابرـاکتشافی (SA-HH)، QMIX و MASSرا بر اساس شاخص زمان کل اجرای وظایف ارائه می‌دهد. داده‌ها نشان می‌دهند که روش MASS در اکثر موارد زمان کمتری برای تکمیل وظایف نسبت به سایر روش‌ها به خود اختصاص داده است. در مقایسه با QMIX، مدل MASS نه‌تنها دارای ساختار غیرمتمرکز و مستقل در سیاست‌های عامل‌ها است، بلکه از پاداش درونی، شــبکه‌های مســتقل و مکانیزم توجه بهره می‌برد که آن را برای محیط‌های پویا با تعداد زیاد عامل، مقیاس‌پذیرتر می‌سازد.

۴,۶. نتایج آموزشی، مقایسه عملکرد و بررسی زمان کل اجرای وظایف

فرآیند آموزشــی مدل‌های MASS و QMIX نشــان می‌دهد که این دو رویکرد در بســیاری از موارد عملکرد مشــابهی در مراحل اولیه آموزش دارند. این شــباهت به‌ویژه در محیط‌هایی که منابع به مقدار کافی موجود اســت و پیچیدگی کمتری دارد، مشهود است. بااین‌حال، هنگامی‌که تعداد منابع یا ایستگاه‌های کاری محدود می‌شود، MASS به دلیل طراحی مبتنی بر مکانیسم انگیزه درونی، برتری قابل‌توجهی نسبت به QMIX پیدا می‌کند اما در شرایطی که منابع محدود است، پایداری و کارایی بیشتری ارائه می‌دهد. این تفاوت ناشی از توانایی MASS در سازگاری سریع با شرایط متغیر و افزایش هماهنگی بین عامل‌ها است. پایداری در آموزش یک ویژگی حیاتی برای الگوریتم‌های یادگیری تقویتی چندعاملی است. این پایداری به معنای توانایی مدل در یادگیری مؤثر و جلوگیری از نوسانات شدید در فرآیند یادگیری است. شکل ۱، نتایج نشان‌دهنده این است که MASS با استفاده از مکانیسم‌های پیشرفته خود، پایداری بیشــتری نســبت به روش‌هایDPR-SPT ، DPR-EDDو SA-HH دارد. این الگوریتم‌ها گرچه در شــرایط ایســتا قابل‌اتکا هستند، اما در شرایط پویا و با ورود وظایف به‌صورت تصادفی دچار افت عملکرد می‌شوند. این مدل با استفاده از مکانیسم‌های انگیزه درونی و هماهنگی بین عامل‌ها توانسته است زمان اجرای وظایف را به حداقل برساند. کاهش زمان اجرای وظایف به‌ویژه در محیط‌های تولیدی با ورود پویا اهمیت بالایی دارد، زیرا تأخیر در تکمیل وظایف می‌تواند منجر به افزایش هزینه‌ها و کاهش بهره‌وری شــود. تحلیل نتایج حاکی از آن اســت که MASS نه‌تنها زمان اجرای وظایف را کاهش می‌دهد، بلکه به دلیل استفاده بهینه از منابع، توانسته است بهره‌وری کلی سیستم را نیز بهبود بخشد. شایان ذکر است که تمام مقادیر زمانی ارائه‌شده در این بخش، از جمله نتایج زمان کل اجرای وظایف، بر حسب دقیقه محاسبه شــده‌اند. یکی از ویژگی‌های منحصــربه‌فرد MASS، مکانیســم پاداش درونی اســت که به عامل‌ها کمک می‌کند تا تصــمیمات بهینه‌تری بگیرند. این مکانیســم از طریق ارائه پاداش‌های مثبت به عامل‌هایی که ســیاســت‌های کارآمدتری اتخاذ می‌کنند و کاهش پاداش برای سیاست‌های ناکارآمد، فرآیند یادگیری را بهبود می‌بخشد بدین صورت عامل‌ها با استفاده از پاداش‌های درونی یاد گرفته‌اند که وظایف خود را به صورت اولویت‌بندی شده و بر اساس نرخ ورود مدیریت کنند.

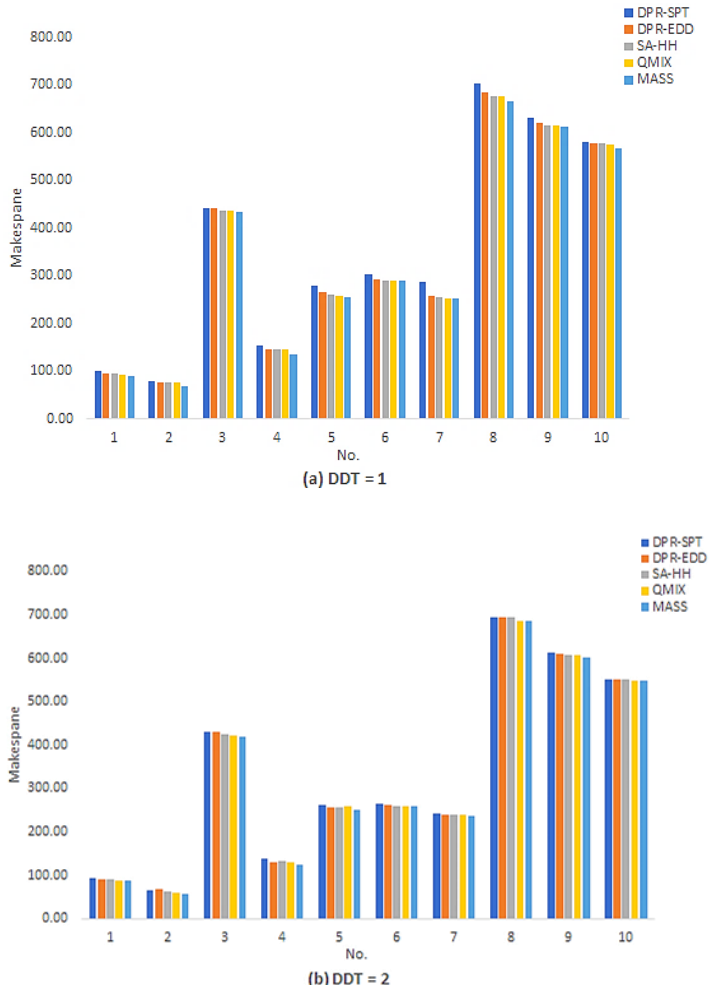
جدول ۳. مقایسه زمان کل اجرای وظایف در مدل‌های مختلف (برحسب دقیقه).

راه‌حل			زمان کل اجرای وظایف									
			D01	D02	D03	D04	D05	D06	D07	D08	D09	D10
PDR	DPR-SPT	DDT = 1	99.91	78.80	442.23	152.40	278.82	301.94	286.59	701.79	631.16	580.83
		DDT = 1.5	92.27	66.14	431.44	139.00	261.50	263.39	242.26	694.06	611.53	550.19
		DDT = 2	89.63	51.77	403.43	127.78	246.51	233.45	201.84	683.19	592.34	537.05
	DPR-EDD	DDT = 1	95.50	76.74	441.93	146.04	264.36	290.94	256.52	685.51	619.45	578.56
		DDT = 1.5	91.48	66.85	430.64	131.10	257.52	262.03	239.29	693.15	608.91	550.42
		DDT = 2	89.90	47.63	402.10	126.11	246.41	231.28	200.47	681.67	589.28	532.48
Hyper-Heuristic	SA-HH	DDT = 1	93.94	76.88	436.87	146.04	260.68	290.65	254.94	675.49	616.10	578.27
		DDT = 1.5	91.34	63.92	424.77	131.67	257.08	259.97	239.84	693.94	607.56	550.45
		DDT = 2	87.91	47.85	401.07	125.66	246.38	231.11	200.26	680.69	588.48	532.72
MADL	QMIX	DDT = 1	93.04	75.20	436.98	144.44	257.16	289.22	252.47	675.69	615.21	575.23
		DDT = 1.5	87.78	59.06	422.17	130.86	257.82	259.53	238.70	687.11	606.55	548.24
		DDT = 2	83.69	43.51	398.12	124.69	244.23	230.17	197.68	680.09	588.03	529.10
	MASS	DDT = 1	90.58	68.74	434.66	135.64	255.97	288.90	252.25	665.53	611.78	568.05
		DDT = 1.5	87.63	58.17	420.16	124.47	250.39	257.73	236.97	686.90	602.34	547.16
		DDT = 2	83.47	43.17	397.64	122.14	243.64	230.64	194.94	676.87	585.09	524.10

مکانیسم توجه در مدل MASS نقش مهمی در بهبود فرآیند یادگیری پاداش درونی دارد. این مکانیسم به عامل‌ها کمک می‌کند تا در هر گام زمانی، وظایف و منابع مرتبط با اولویت بیشتری را شناسایی کنند. این موضوع باعث شده است که عامل‌ها بتوانند با تمرکز بیشتر بر وظایف حیاتی، تصمیمات مؤثرتری بگیرند. استفاده از این مکانیسم کارایی مدل را افزایش داده و توانایی آن را در مدیریت وظایف پیچیده بهبود بخشیده است.

۵. نتیجه‌گیری

در این پژوهش، یک راهکار یادگیری تقویتی چندعامله با بهره‌گیری از انگیزش درونی برای حل مسائل زمان‌بندی پویا در کارگاه‌های تولیدی منعطف ارائه شده است. سیستم پیشنهادی از یک فرآیند آموزش متمرکز بهره می‌برد که امکان اجرای غیرمتمرکز زمان‌بندی را فراهم می‌سازد. در این رویکرد، زمان‌بندی وظایف در یک محیط چندعامله با استفاده از یک فرآیند تصمیم‌گیری مارکوف غیرمتمرکز انجام می‌شود که هدف آن ایجاد تعادل میان هماهنگی عامل‌ها و استقلال عملکرد هر عامل است. نتایج حاصل از این مطالعه نشان می‌دهد که سیستم چندعامله مبتنی بر انگیزش درونی ما، در مقایسه با روش‌های متداول، کارایی بهتری در بهینه‌سازی زمان تکمیل کارها دارد. MASS در مراحل پیشرفته آموزش عملکرد پایدارتر و بهتری از خود نشان می‌دهد. این عملکرد به‌ویژه در نمونه‌های آزمایشی D1-D10 قابل‌مشاهده است. این روش قادر است در یک چارچوب یادگیری تقویتی چندعامله ، ورود پویا و غیرقابل پیش‌بینی کارها را مدیریت کند، بدون اینکه پایداری فرآیند یادگیری مختل شود. علاوه بر بهبود کارایی و کاهش زمان تکمیل، مدل پیشنهادی انعطاف‌پذیری بالایی را در شرایط پویا نشان داده و پایداری فرایند آموزش آن تأیید شده است. همچنین، ساختار پیشنهادی این مدل با توجه به طراحی ماژولار و معماری غیرمتمرکز عامل‌ها، قابلیت تعمیم به سایر مسائل زمان‌بندی نظیر Flow Shop و Open Shop را نیز داراست، به شرط بازتعریف مناسب توالی عملیات و محدودیت‌های منابع در محیط آموزش.



شکل۱. مقایسه روند آموزش مدل‌ها در مقادیر مختلف DDT

این ویژگی‌ها نشان می‌دهند که این مدل می‌تواند برای کاربردهای صنعتی گوناگون در حوزه زمان‌بندی تولید، از جمله صنایع پیچیده و محیط‌های تولیدی با تغییرات مداوم، مورد استفاده قرار گیرد. هرچند در مسیر پیاده‌سازی صنعتی مدلMASS ، چالش‌هایی از جمله نیاز به داده‌های دقیق از محیط واقعی، تطبیق با سیستم‌های کنترلی موجود، مقاومت در برابر نویز محیطی، و آموزش مجدد در صورت تغییرات ساختاری وجود خواهد داشت. با این حال، با توجه به طراحی غیرمتمرکز و قابلیت انعطاف‌پذیری مدل، این چالش‌ها قابل مدیریت و حل هستند.

References

[۱]Aubret, A., Matignon, L., & Hassas, S. (2023). *An information-theoretic perspective on intrinsic motivation in reinforcement learning: A survey*. Entropy, 25(2), 327. https://www.mdpi.com/1099-4300/25/2/327.

[۲]Borboni, A., Reddy, K. V. V., Elamvazuthi, I., AL-Quraishi, M. S., Natarajan, E. & ,Azhar Ali, S. S. (2023). *The expanding role of artificial intelligence in collaborative robots for industrial applications: a systematic review of recent works*. Machines, 11(1), 111. https://www.mdpi.com/2075-1702/11/1/111.

[۳]Brandimarte, P. (1993). *Routing and scheduling in a flexible job shop by tabu search*. Annals of Operations research, 41(3), 157-183. https://doi.org/10.1007/BF02023073.

[٤]Du, Y., Han, L., Fang, M., Liu, J., Dai, T., & Tao, D. (2019). *Liir: Learning individual intrinsic reward in multi-agent reinforcement learning*. Advances in neural information processing systems, 32. https://proceedings.neurips.cc/paper_files/paper/2019/file/07a9d3fed4c5ea6b17e80258dee231fa-Paper.pdf.

[٥]Hao, J., Yang, T., Tang, H., Bai, C., Liu, J., Meng, Z., Liu, P., & Wang, Z. (2023). *Exploration in deep reinforcement learning: From single-agent to multiagent domain*. IEEE Transactions on Neural Networks and Learning Systems. https://ieeexplore.ieee.org/document/10021988.

[٦]Johnson, D., Chen, G., & Lu, Y. (2022). *Multi-agent reinforcement learning for real-time dynamic production scheduling in a robot assembly cell*. IEEE Robotics and Automation Letters, 7(3), 7684-7691. https://ieeexplore.ieee.org/document/9801608.

[۷]Hoseini, F., Sepehrzadeh, H., & Kheyri, M. (202٤).*Presenting an" Adaption Ahead" Optimization Algorithm for Training Models Used in Deep Learning*. Karafan Journal. https://karafan.nus.ac.ir/article_210487.html.

[۸]Keung, K. L., Lee, C. K., Liqiao, X., Chao, L., Bufan, L., & Ji, P. (2023). *A cyber-physical robotic mobile fulfillment system in smart manufacturing: The simulation aspect*. Robotics and Computer-Integrated Manufacturing, 83, 102578. https://doi.org/10.1016/j.rcim.2023.102578.

[۹]Ladosz, P., Weng, L., Kim, M., & Oh, H. (2022). *Exploration in deep reinforcement learning: A survey*. Information Fusion, 85, 1-22. https://doi.org/10.1016/j.inffus.2022.03.003.

[۱۰]Lim, K. C. W., Wong, L.-P., & Chin, J. F. (2023). *Hyper-heuristic for flexible job shop scheduling problem with stochastic job arrivals*. Manufacturing letters, 36, 5-8. https://doi.org/10.1016/j.mfglet.2022.12.009.

[۱۱]Liu, C.-L., Chang, C.-C., & Tseng, C.-J. (2020). *Actor-critic deep reinforcement learning for solving job shop scheduling problems*. Ieee Access, 8, 71752-71762. https://ieeexplore.ieee.org/document/9066984.

[۱۲]Rafiei, S. H., Ojaghi, M., & Sabouri, M. (2023). *The Effective Use of SVM to Detect Faults in Electric Machines by Finite Element Analysis*. Karafan Journal, 20(3), 367-392. https://karafan.tvu.ac.ir/article_181325.html?lang=en.

[۱۳]Oroojlooy, A., & Hajinezhad, D. (2023). *A review of cooperative multi-agent deep reinforcement learning*. Applied Intelligence, 53(11), 13677-13722. https://link.springer.com/article/10.1007/s10489-022-04105-y.

[۱٤]Sun, Y., Chung, S.-H., Wen, X., & Ma, H.-L. (2021). *Novel robotic job-shop scheduling models with deadlock and robot movement considerations*. Transportation Research Part E: Logistics and Transportation Review, 149, 102273. https://doi.org/10.1016/j.tre.2021.102273.

[۱۵]Benhari, M., & Hosseini, R. (2024). *An Intelligent Ensemble Model of Uncertainty Management in Belief Network for the Classification of Microscopic Images to Detect Cervical Cancer*. Karafan Journal, 2۱(۱), 91-110. https://karafan.nus.ac.ir/article_181335.html?lang=en.

[۱٦]Wang, M., Zhang, J., Zhang, P., Cui, L., & Zhang, G. (2022). *Independent double DQN-based multi-agent reinforcement learning approach for online two-stage hybrid flow shop scheduling with batch machines*. Journal of Manufacturing Systems, 65, 694-708. https://doi.org/10.1016/j.jmsy.2022.11.001.

[۱۷]Xiong, W., & Fu, D. (2018). *A new immune multi-agent system for the flexible job shop scheduling problem*. Journal of Intelligent Manufacturing, 29, 857-873. https://link.springer.com/article/10.1007/s10845-015-1137-2.

[۱۸]Xu, M., Zhang, F., Mei, Y., &Zhang, M. (2022). *Genetic programming with multi-case fitness for dynamic flexible job shop scheduling*. 2022 IEEE Congress on Evolutionary Computation (CEC). https://ieeexplore.ieee.org/document/9870340.

[۱۹]Yang, S., Wang, J., & Xu, Z. (2022). *Real-time scheduling for distributed permutation flowshops with dynamic job arrivals using deep reinforcement learning*. Advanced Engineering Informatics, 54, 101776. https://doi.org/10.1016/j.aei.2022.101776.

[۲۰]Zhang, J.-D., He, Z., Chan, W.-H., & Chow, C.-Y. (2023). *DeepMAG: Deep reinforcement learning with multi-agent graphs for flexible job shop scheduling*. Knowledge-Based Systems, 259, 110083. https://doi.org/10.1016/j.knosys.2022.110083.

[۲۱]Zhang, Y., Zhu, H., Tang, D., Zhou, T., & Gui, Y. (2022). *Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems*. Robotics and Computer-Integrated Manufacturing, 78, 102412. https://doi.org/10.1016/j.rcim.2022.102412.