Journal of System Simulation

Volume 35 | Issue 11 Article 5

11-30-2023

# Intercell Dynamic Scheduling Method Based on Deep Reinforcement Learning

Jing Ni
*University of Shanghai for Science and Technology, Shanghai 200093, China*, nijing501@126.com

Mengke Ma
*University of Shanghai for Science and Technology, Shanghai 200093, China*

# Intercell Dynamic Scheduling Method Based on Deep Reinforcement Learning


## Abstract

Abstract: In order to solve the intercell scheduling problem of dynamic arrival of machining tasks and realize adaptive scheduling in the complex and changeable environment of the intelligent factory, a scheduling method based on a deep Q network is proposed. A complex network with cells as nodes and workpiece intercell machining path as directed edges is constructed, and the degree value is introduced to define the state space with intercell scheduling characteristics. A compound scheduling rule composed of a workpiece layer, unit layer, and machine layer is designed, and hierarchical optimization makes the scheduling scheme more global. Since double deep Q network (DDQN) still selects sub-optimal actions in the later stage of training, a search strategy based on the exponential function is proposed. Through simulation experiments of different scales, it is verified that the proposed method can deal with the changeable dynamic environment and quickly generate an optimal scheduling scheme.



## Keywords

intercell scheduling, dynamic scheduling, reinforcement learning, degree value, compound rule

第 35 卷第 11 期 系统仿真学报© Vol. 35 No. 11
2023 年 11 月 Journal of System Simulation Nov. 2023

# 基于深度强化学习的跨单元动态调度方法

倪静，马梦珂
(上海理工大学，上海 200093)

**摘要：**为解决加工任务动态到达的跨单元调度问题，使其能够在智能车间复杂多变的环境中实现自适应调度，提出一种基于深度Q网络的调度方法。构建以单元为节点，工件跨单元加工路径为有向边的复杂网络，引入度值定义了具有跨单元调度特征的状态空间。设计了由工件层、单元层和机器层组成的复合调度规则，分层优化使调度方案更加全局化。针对DDQN(double deep Q networks)在训练后期还会选择次优动作的问题，提出了以指数函数为主体的搜索策略。通过不同规模的仿真实验，验证了所提方法能够应对多变的动态环境，快速生成较优的调度方案。

**关键词：**跨单元调度；动态调度；强化学习；度值；复合规则
中图分类号：TP18；TP391 文献标志码：A 文章编号：1004-731X(2023)11-2345-14
DOI: 10.16182/j.issn1004731x.joss.22-0666
**引用格式:** 倪静, 马梦珂. 基于深度强化学习的跨单元动态调度方法[J]. 系统仿真学报, 2023, 35(11): 2345-2358.
**Reference format:** Ni Jing, Ma Mengke. Intercell Dynamic Scheduling Method Based on Deep Reinforcement Learning [J]. Journal of System Simulation, 2023, 35(11): 2345-2358.

## Intercell Dynamic Scheduling Method Based on Deep Reinforcement Learning

*Ni Jing*, *Ma Mengke*
(University of Shanghai for Science and Technology, Shanghai 200093, China)

**Abstract:** In order to solve the intercell scheduling problem of dynamic arrival of machining tasks and realize adaptive scheduling in the complex and changeable environment of the intelligent factory, a scheduling method based on a deep Q network is proposed. *A complex network with cells as nodes and workpiece intercell machining path as directed edges is constructed, and the degree value is introduced to define the state space with intercell scheduling characteristics. A compound scheduling rule composed of a workpiece layer, unit layer, and machine layer is designed, and hierarchical optimization makes the scheduling scheme more global. Since double deep Q network (DDQN) still selects sub-optimal actions in the later stage of training, a search strategy based on the exponential function is proposed.* Through simulation experiments of different scales, it is verified that the proposed method can deal with the changeable dynamic environment and quickly generate an optimal scheduling scheme.

**Keywords:** intercell scheduling; dynamic scheduling; reinforcement learning; degree value; compound rule

## 0 引言

在当今智能制造新模式下，生产制造的全过程表现为柔性生产、个性化服务和高端定制化，越来越多的工艺需要特殊的机器设备加工，而多个单元协同生产能够避免因购置机器产生成本剧增的情况。跨单元调度问题研究可以优化加工任务在多个单元的全局调度方案，从而更好地提高

收稿日期：2022-06-20 修回日期：2022-09-04
基金项目：教育部人文社会科学基金(19YJAZH064)
第一作者：倪静(1972-)，女，副教授，博士，研究方向为在线社会网络，智能优化算法。E-mail：nijing501@126.com

生产效率。此外不确定性因素导致此类调度方案的适应性差，需要实时调整。因此，生产车间迫切需要一种实时在线的调度方法来处理动态跨单元调度问题，从而使得生产加工过程更加高效且稳定地运行。

大多数跨单元调度方法通常将实际问题转化为带约束的组合优化问题来实现目标的最大化。文献[1]针对早期研究仅考虑企业内部的单元调度而忽视了单元间运输时间的问题，使用基于拍卖模型的启发式方法优化了包含运输时间的多个单元协作生产的完工时间。文献[2]提出了一种基于蚁群优化算法的信息素模型，在单元间路径决策过程中考虑了机器加工能力、机器的释放时间和加工队列大小，以及工件在单元间移动时间，使得调度方案比仅仅考虑单个单元更加全局化。文献[3]将跨单元调度问题转化为析取图调度模型，并使用改进的和声搜索算法求解。文献[4]考虑了具有与序列相关的单元设置时间和单元间运输时间约束的单元调度问题。文献[5]考虑到跨单元调度中的机器和人力资源约束，构建了单元划分与调度联合决策模型，提出离散细菌觅食算法求解以减少制造成本。文献[6]研究了运输空间约束下的多单元协作调度问题。文献[7]通过提升运输中的车辆利用率来降低跨单元调度的运输成本和完工时间。上述文献考虑跨单元调度的多种实际因素，使用启发式算法多次迭代求得较优的调度方案。

但是启发式算法无法快速响应生产实际中的动态环境，难以满足智能车间实时调度的需求。而机器学习能够实现在动态环境下的自适应调度，调度智能体通过经验学习建立多个状态和目标之间的关系，在不同的生产环境中适应性地执行最优的调度动作。文献[8]考虑机器故障环境下的柔性动态调度问题，使用Q学习算法使得智能体能够在不同的机器故障状态下快速为工件安排新的机器以实现自适应动态调度。针对Q学习算法在解决调度问题时因为算例规模增加而引起Q表格维度灾难问题，文献[9]将状态空间通过聚类算法划分成不同的类来减少状态的维度，为了降低聚类后状态与实际状态之间的误差，将Q值更新结合状态差异度以提高算法的准确性。文献[10]解决了工件随机到达的动态柔性作业车间调度问题，以最小化延迟时间为目标定义了7种状态和6种复合调度，使用神经网络代替Q表格更新的DQN (deep Q network)算法求解，提高了自适应调度算法在动态制造环境下的性能。文献[11]针对具有交货期约束的动态调度问题，设计了5个连续的状态特征输入DNN(deep nueral network)网络，并选择了10个启发式调度规则作为DQN的动作集，实验结果表明DQN算法在大规模算例中求性能优于Q学习算法。文献[12]提出了一种基于多智能体的DQN算法，其中一个智能体负责工件的选择，而另一个负责向机器分配作业，实验表明DQN能够在相对较少的训练下实时计算出高质量的解决方案。除此之外，DQN算法求解工件加工时间不确定[13]、紧急插单[14]的动态问题上也表现出较好的性能。上述研究表明深度Q网络算法能够快速响应不同生产环境中的动态因素。

尽管深度Q网络算法在动态调度问题上表现出较好的性能，但是在求解动态跨单元调度问题上缺乏相关的应用，使用其求解问题的关键是定义调度动作和环境状态。①在动作空间的设计方面，启发式规则操作简单不需要建模和调整参数，多数研究将调度规则设计为强化学习的动作空间，取得了较好的调度结果。例如文献[15]将最短加工时间、最短设置时间、最小队列长度和最小期望队列时间4种传统的调度规则封装到Q学习算法中，实验表明求解结果优于单一调度规则和遗传算法。但单一的规则仅定义工件的优先级，难以适用于更为复杂的调度环境。而复合调度规则在复杂调度环境中具有更好的性能[16]，对此文献[17]提出一种同时分配机器和工序的复合调度规则，先对工序进行优先级定义，再为工序选择加工机器。文献[18]提出了运用数据包络分析方法来确定




DOI: 10.16182/j.issn1004731x.joss.22-0666

基本调度规则的相对权重，最终进行线性组合设计成求解效率较好的复合规则。然而上述复合调度规则并不能直接用于求解跨单元调度问题，因此设计符合问题特征的复合规则是求解的关键一步。②在状态空间的定义方面，不同的调度属性被定义为环境状态。文献[19]使用了加工时间、调度结果和机器利用率来定义柔性车间调度的环境状态从而取得了较好的调度结果。文献[20]将工序作为网络节点并将网络度值作为调度状态，实验表明复杂网络特征能将调度方案转化为训练数据集使得调度结果优于已有的调度规则。单元调度问题作为一种更复杂的调度系统，利用网络特征来描述环境的状态，能够提供更多的训练数据从而提升算法的学习效率。

综上所述，现有文献对于动态跨单元调度问题的研究较少。区别于传统的动态车间调度问题，跨单元调度由于加工任务需要在多个单元内加工，需要在调度决策时刻充分考虑单元间的运输时间等因素从而使调度方案更加全局化。本文将具有跨单元调度特征的9种状态和18种复合调度规则封装到DDQN算法中，在工件到达时间不确定的情况下实现自适应调度。为了更全面地描述动态调度系统的状态变化，同时考虑到复杂网络结构具有动态随机的特点，多变的网络特征能够扩充状态表达的维度，本文以加工单元作为网络节点构建复杂网络，引入度值定义全局的跨单元状态和调度动作，对现有的调度属性进行补充。

# 1 动态跨单元调度问题描述

动态跨单元调度问题定义为：$n$个待加工工件的集合为$i=\{1,2\cdots,n\}$，工件$i$的工序集合为$j=\{1,2\cdots,n_i\}$，$m$台机器集合$k=\{1,2\cdots,m\}$，$h$个制造单元集合为$u=\{1,2,\cdots,h\}$。需满足的约束条件有：①工件具有随机的到达时间且加工全过程至少需要由2个制造单元联合加工；②工序每次调度只能选择一个制造单元中的一台加工机器；③工序开始加工后不能中断；④工序的开始加工时间不能小于紧前工序的完工时间与运输时间之和；⑤同一工件的工序具有固定的加工顺序。

本文的调度方案是将工件分配到合适的制造单元和加工机器，并确定机器上的加工顺序，以实现最大完工时间最小化。目标函数如下：

$$\min F=\max t_i \tag{1}$$

式中：$t_i$为最后一个加工工件的完工时间。

# 2 基于MDP的跨单元调度系统

动态跨单元调度问题可以看作一个调度智能体在不同的调度时刻依次为待加工的工件选择制造单元和机器的顺序决策过程，一个完整的调度周期是所有的工件都被调度。为了使得调度智能体实现自适应决策，首要步骤是将具体问题转化为一个MDP(Markov decision process)问题，即定义与智能体交互的环境状态、调度智能体可执行的动作以及奖励函数。

## 2.1 符号定义

下面将所涉及的各种变量，一一进行解释。见表1，2。

(1) 索引

表1 符号定义
Table 1 Symbol definition

| 变量 | 含义 |
|---|---|
| $i$ | 工件编号 |
| $n$ | 工件数量 |
| $k$ | 机器编号 |
| $m$ | 机器数量 |
| $j$ | 工序编号 |
| $n_i$ | 工件$i$的工序数量 |
| $u$ | 制造单元编号 |
| $h$ | 制造单元数量 |
| $v$ | 不同单元间的运输速度 |

(2) 系统变量

表2 系统变量
Table 2 System variables

| 变量 | 含义 |
|---|---|
| $o_{ij}$ | 工件$i$的第$j$道工序 |
| $T_{ri}$ | 工件$i$到达调度系统的时间 |
| $T_{si}$ | 工件$i$的开始加工时间 |
| $T_{sk}$ | 机器$k$的开始加工时间 |
| $T_{su}$ | 单元$u$最早开始加工时间 |
| $z_u$ | 单元$u$内归置的机器数量 |
| $T_{pijk}$ | 工序$o_{ij}$在机器$k$上的加工时间 |
| $T_{oij}$ | 工序$o_{ij}$的加工结束时间 |
| $d_{uu'}$ | 不同单元之间的运输距离 |
| $a_{ijk}$ | 工序$o_{ij}$可以在机器$k$上加工则为1，否则为0 |
| $b_{ku}$ | 机器$k$归置于单元$u$则为1，否则为0 |
| $OP_i(t)$ | $t$时刻工件$i$已安排加工的工序数量 |
| $U_u(t)$ | $t$时刻单元$u$的利用率 |
| $U_k(t)$ | $t$时刻机器$k$的利用率 |
| $T_{\text{ave}}(t)$ | $t$时刻平均工件等待时间 |
| $T_{cu}(t)$ | $t$时刻单元$u$的最大完工时间 |
| $T_{ck}(t)$ | $t$时刻机器$k$的最大完工时间 |

(3) 决策变量

$x_{ijk}$:工序$o_{ij}$在机器$k$上加工则为1，否则为0；$q^i_{ju}$:工序$o_{ij}$在单元$u$上加工，否则为0。

## 2.2 状态空间

状态空间的定义需要准确描述调度环境的特征，在跨单元调度问题中，环境主体包含机器、工件及制造单元，定义如下9种状态用于描述系统环境。

(1) 平均制造单元利用率$UC_{\text{ave}}$描述了制造单元的加工负荷平均水平。具体表示为

$$UC_{\text{ave}}(t)=\frac{\sum_{u=1}^{h}U_u(t)}{h} \tag{2}$$

$$U_u(t)=\frac{\sum_{k=1}^{z_u}\sum_{i=1}^{n}\sum_{j=1}^{OP_i(t)}T_{pijk}\cdot x_{ijk}\cdot b_{ku}}{z_u\cdot(T_{cu}(t)-T_{su})} \tag{3}$$

(2) 单元利用率方差描述了制造单元的加工负荷整体差异，具体表示为

$$UC_{\text{avr}}(t)=\sqrt{\frac{\sum_{u=1}^{h}(U_u(t)-UC_{\text{ave}}(t))^2}{h}} \tag{4}$$

(3) 平均网络度值$K_{\text{ave}}(t)$描述了以单元节点的网络连接紧密程度。

跨单元制造系统具有明显的复杂网络特征，将跨单元调度中的制造单元作为网络节点，网络模型可表达为$G=(V_u,E_u)$，节点$V_u$由制造单元$u$组成，有向边$E_u$为单元节点间的有向边，如图1所示。工序$o_{11}$在$u_1$加工结束后运输到$u_2$进行加工即生成一条网络边。复杂网络结构的动态性体现在网络中节点之间的关联变化引起网络边的出现和消失，从而动态变化的网络特征为状态定义提供了新的数据支撑。节点度值描述了全局网络节点间联系紧密的程度，在加工网络中有向边表示工件的跨单元加工路径，因此节点度值的大小与工件跨单元加工次数成正比关系。平均网络度值为

$$K_{\text{ave}}(t)=\frac{1}{h}\sum_{V=1}^{h}k_V \tag{5}$$

式中：$k_V$为节点$V$的度值，$k_V=k_{\text{out}}+k_{\text{in}}$，其中$k_{\text{out}}$为出度；$k_{\text{in}}$为入度。

(4) 平均机器利用率$U_{\text{ave}}(t)$描述了当前工件分配在各机器上的工作负荷，具体表示为

$$U_{\text{ave}}(t)=\frac{\sum_{k=1}^{m}U_k(t)}{m} \tag{6}$$

$$U_k(t)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{OP_i(t)}T_{pijk}\cdot x_{ijk}}{T_{ck}(t)-T_{sk}} \tag{7}$$

(5) 机器利用率方差描述了机器负荷之间的整体差异，具体表示为

$$U_{\text{avr}}(t)=\sqrt{\frac{\sum_{m=1}^{k}(U_k(t)-U_{\text{ave}}(t))^2}{m}} \tag{8}$$

(6) 平均机器空闲时间描述了所有机器空闲时间的平均值，具体表示为

$$MF_{\text{ave}}(t)=\frac{\sum_{k=1}^{m}CT_k(t)-s_k-\sum_{i=1}^{n}\sum_{i=1}^{OP_i(t)}p_{ijk}\cdot x_{ijk}}{m} \tag{9}$$




DOI: 10.16182/j.issn1004731x.joss.22-0666

(7) 平均工件完成率 $CR_{\text{ave}}(t)$ 描述了工件的加工进度为

$$CR_{\text{ave}}(t)=\frac{1}{n}\sum_{i=1}^{n}\frac{OP_i(t)}{n_i} \tag{10}$$

(8) 工件等待时间方差 $T_{i\text{avr}}(t)$ 描述所有工件从到达到开始加工过程中的等待时长的体差异。

$$T_{i\text{avr}}(t)=\sqrt{\frac{\sum_{i=1}^{n}(T_{si}-Tr_i-T_{i\text{ave}}(t))^2}{n}} \tag{11}$$

式中：$T_{i\text{ave}}(t)$ 为 $t$ 时刻平均工件等待时间，计算公式为

$$T_{i\text{ave}}(t)=\frac{\sum_{i=1}^{n}Ts_i-Tr_i}{n} \tag{12}$$

(9) 等待加工的所有工件中剩余加工时间的最小值与所有工件中剩余加工时间的最大值的比

$$w(t)=\frac{\min\limits_{r_i\le t}\sum\limits_{j=OP_i(t)+1}^{n_i}T_{p\text{ave}}}{\max\limits_{r_i\le t}\sum\limits_{j=OP_i(t)+1}^{n_i}T_{p\text{ave}}} \tag{13}$$

式中：$T_{pave}$ 为工序平均加工时间，计算公式为

$$T_{p\text{ave}}=\frac{\sum_{k=1}^{m}a_{ijk}\cdot T_{pijk}}{\sum_{k=1}^{m}a_{ijk}} \tag{14}$$

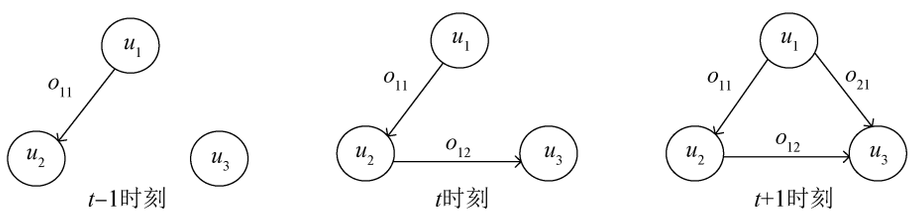



图1 不同时刻加工网络示意图
Fig. 1 Machining network at different time

## 2.3 动作空间

动作空间是智能体可以执行的所有动作组成的集合。智能体通过经验学习建立多个调度状态和目标之间的关系，使得不同的调度规则可以被适应性地选择，通过执行某个调度规则来选择合适的工件或机器。

跨单元调度的动作涉及到制造单元和机器的选择，因此本文将跨单元调度决策过程依次拆分为加工工件分配加工单元和加工机器的过程，设计了工件、制造单元和加工机器 3 个层面的优先级规则，将 3 种规则排列组合为 18 种复合调度动作，其组合方式，如图 2 所示。

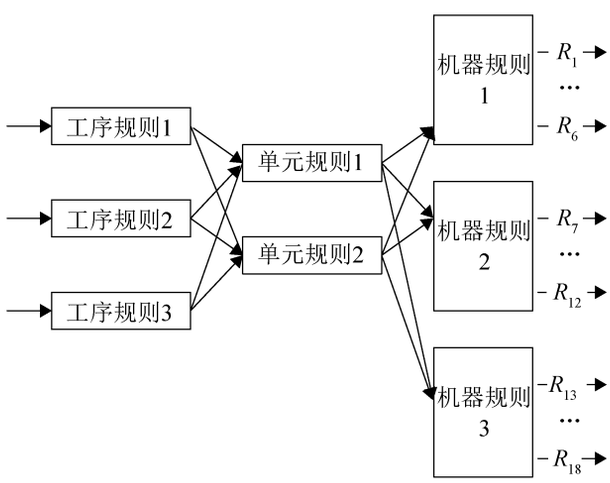



图2 复合调度规则组合方式
Fig. 2 Compound scheduling rule combination

### 2.3.1 工序调度规则

调度智能体根据当前环境的状态，在待加工的工件集中对待分配的工序计算优先级，再择优选择加工单元和加工机器完成调度任务。本文选取“最小化完工时间”这一性能指标下常用的 3 种启发式调度规则。

(1) 优先选择剩余加工时间最短的工件加工，以期利用最短的时间提高全局的工件完成率。各



个待加工件的优先级表达式为

$$R_{ij}=\arg\min\sum_{j=OP_i(t)+1}^{n_i}\bar{T}_{pij} \tag{15}$$

(2) 优先选择工件完成率较低的工件加工，避免工件一直等待被加工的情况。各个待加工工件的优先级为

$$R_{ij}=\arg\min\frac{OP_i(t)}{n_i} \tag{16}$$

(3) 优先选择当前工序加工时间与总加工时间比值最大的工件加工，较早加工最耗时的工序以提高后期调度决策的灵活性。

$$R_{ij}=\arg\max\frac{T_{pave}}{\sum_{j=1}^{n_i}T_{pave}} \tag{17}$$

### 2.3.2 制造单元调度规则

制造单元的分配主要受到2个因素的影响：①调度时刻的单元利用率，单元利用率相差较小可以认为全局的制造负荷均衡；②调度时刻单元节点的网络度值，通过度值来描述单元的加工效率，使得调度方案更加全局化。

(1) 选择单元利用率最低的单元加工，以期通过该动作平衡全局单元的加工负荷，避免出现某个单元一直闲置的情况。各个加工单元的优先级为

$$R_u=\arg\min U_u(t) \tag{18}$$

(2) 优先选择入度与出度的差值最小的单元加工。各个待加工工件的优先级为

$$R_u=\arg\min f_u(t) \tag{19}$$

如图1所示网络图，在调度决策时刻单元节点入度表示当前制造单元上等待加工的工序数，节点出度表示加工完成待运输的工序数量，入度与出度的差值越小，即制造单元内堆积的加工工件越少。如式(20)定义$f_u(t)$为入度与出度的差值，优先选择值差值最小的加工单元加工，以期通过该动作选择较为空闲的加工单元，协调全局的加工效率。定义二进制矩阵$\boldsymbol{Q}^i=\left\{q_{ju}^i\right\}$，在$\boldsymbol{Q}^i$中，每行表示工件$i$的工序$j$，每列表示单元$u$，$q_{j,u}^i=1$表示工序$o_{ij}$在单元$u$上加工，否则为0；因此节点上0到1的变化表示节点入度，1到0的变化表示节点出度。

$$f_u(t)=\sum_{i=1}^{n}\sum_{j=2}^{n_i}(q_{j,u}^i(1-q_{j-1,u}^i)-q_{j-1,u}^i(1-q_{j,u}^i)) \tag{20}$$

### 2.3.3 机器调度规则

确定加工单元后，在该制造单元内对可加工的机器进行排序，最终确定加工机器。

(1) 优先选择最早可用的机器

在机器分配中，优先选择最早可用的机器是一重要规则，能够较大程度减少机器的空闲时间，缩短完工时间。算法1在调度决策时刻考虑到跨单元加工的工件运输时间从而避免工件到达后长期等待和未到达之前机器空闲的情况。

算法1　最早可用机器

输入：工序$o_{ij}$，单元坐标$(X,Y)$，$CT_k(t)$

1：$Able_k(t)\leftarrow\{k|a_{ijk}=1\}$

2：if $j$=1 then

3：　$k\leftarrow\arg\min_{k\in Able_k(t)}CTk(t),r_i$

4：else

5：　for $k=1{:}m$ do

6：　　$S_{ij}\leftarrow C_{i,j-1}+\frac{duu'}{40}$

7：　　$k\leftarrow\arg\min_{k\in Able(t)}((S_{ij}-CT_k(t)),0)$

8：　end for

9：end if

10：assgin $O_{ij}$ on $k$

(2) 优先选择机器利用率最低的机器，平衡机器负荷，提高机器的利用率。各个加工机器为

$$R_k=\arg\min U_k(t) \tag{21}$$

(3) 优先选择加工能力较低的机器，加工能力定义为该机器可以加工的工序总数。当机器的加工能力定义为该机器可以加工的工序总数。当机器的加工能力较低时，工件选择该机器的可能性较小，因此该动作可以尽快抢占此类机器，避免机器一直闲置的情况。各个加工机器的优先级为

$$R_k=\arg\min\left(\sum_{i=1}^{n}\sum_{j=1}^{n_i}a_{ijk}\right) \tag{22}$$




DOI: 10.16182/j.issn1004731x.joss.22-0666

### 2.4 奖励函数

回报函数是指对智能体执行的每一个决策给出反馈，以引导智能体有效的学习。本文的回报函数设置为即时奖励，即每一次执行动作都会给予反馈，一个完整调度方案的累积奖励是即时奖励之和。文献[21]中证明了机器利用率与完工时间紧密相关，因此本文设置的奖励函数为平均机器利用率，而奖励值过高或过低会使梯度更新慢影响算法速度。根据贝尔曼公式，奖励值加减一个非零常数，会破坏环境本身的奖励设置，让$Q$值变成一个受执行步数影响的值，因此使用奖励缩放的方法设置系数$\lambda$用于调整大小，奖励函数计算如下：

$$R_t=\lambda U_{\text{ave}}(t) \tag{23}$$

## 3 DDQN 算法

DDQN（double DQN）是基于Q学习和DQN的一种改进算法。Q学习是基于价值函数的强化学习算法，通过构建Q表格来存储状态–动作的期望值，训练好的Q表格用于指导算法选择最优的调度规则。$Q$值更新为

$$Q(s_t,a_t)=Q(s_t,a_t)+\alpha[r_t+\gamma\max_{a_{t+1}} Q(s_{t+1},a_{t+1})-Q(s_t,a_t)] \tag{24}$$

式中：$\alpha$为学习率；$\gamma$为折扣因子。

针对Q学习算法因为状态空间的增大在更新Q表时造成的维度灾难问题，DQN使用神经网络代替Q表格，较好地解决了该问题。同时增加了经验池来存储数据，通过小批量样训练打破了数据间的相关性,提升了智能体的学习效率。DQN使用2个结构相同的Q网络计算目标$Q$值和当前$Q$值，训练目标$Q$值与当前$Q$值的损失函数来更新网络参数$\theta$，第$i$个损失函数为

$$L_i(\theta_i)=E[(y_i-Q(s,a;\theta_i))^2] \tag{25}$$

式中：$y_i$为目标$Q$值。

虽然DQN算法通过贪婪策略计算目标$Q$值可以快速让$Q$值收敛于最优，但也容易导致过度估计，最终导致整个网络无法收敛。因此DDQN采用了将动作选择与策略评估分开，使用了两个独立的动作价值估计函数，使得$Q$值更加接近真实值。其更新目标如式(26)所示，不再直接在目标Q网络中找各个动作的最大$Q$值，而是现在当前Q网络中找出最大$Q$值对应的动作，然后利用这个动作在目标Q中计算目标$Q$值。

$$Y_t^{\text{double}Q}=R_{t+1}+\gamma Q(s_{t+1},\arg\max Q(s_{t+1},a;\theta_t);\theta_t^-) \tag{26}$$

在动态跨单元制造环境下，加工任务随客户的订单需求不断进入调度系统。调度环境由机器、工件和制造单元组成，将复合调度规则作为动作，平均机器利用率作为奖励。具体的模型框架如图3所示，智能体将9种环境状态输入神经网络，网络经过训练输出不同状态动作对的$Q$值，接着执行$Q$值最大的调度规则确定下一步加工的工件，以及对应的加工单元和机器。机器按照工件的分配顺序依次加工，完成后将当前环境的状态和奖励反馈给智能体，并且不断重复此过程。其中经验池用来存储训练样本，并用于网络的训练。由此DDQN算法得到了在不同状态下采取不同规则的最优策略。

### 3.1 神经网络结构

根据2.2和2.3节中设计的调度状态和动作，本文构造了2个结构相同的BP神经网络。如图3所示，神经网络由输入层、隐藏层、输出层三部分组成，输入层神经元分别对应着状态所有特征向量，输出层神经元对应着动作空间，计算输出每一个状态–动作对的价值$Q(s,a)$。神经网络的输入节点数为9，输出节点数为18，采用RELU激活函数。

### 3.2 经验回放

为了进行经验回放，DDQN将智能体与环境进行交互的经验数据$\{s_t,a_t,r_t,s_{t+1}\}$存储到经验池，并在训练时对经验池$D$中的经验数据进行随小批



量采样更新深度神经网络的参数，将过去的经验数据和目前的经验数据混合，打破了数据的相关性及非静态分布问题，算法收敛的更快。

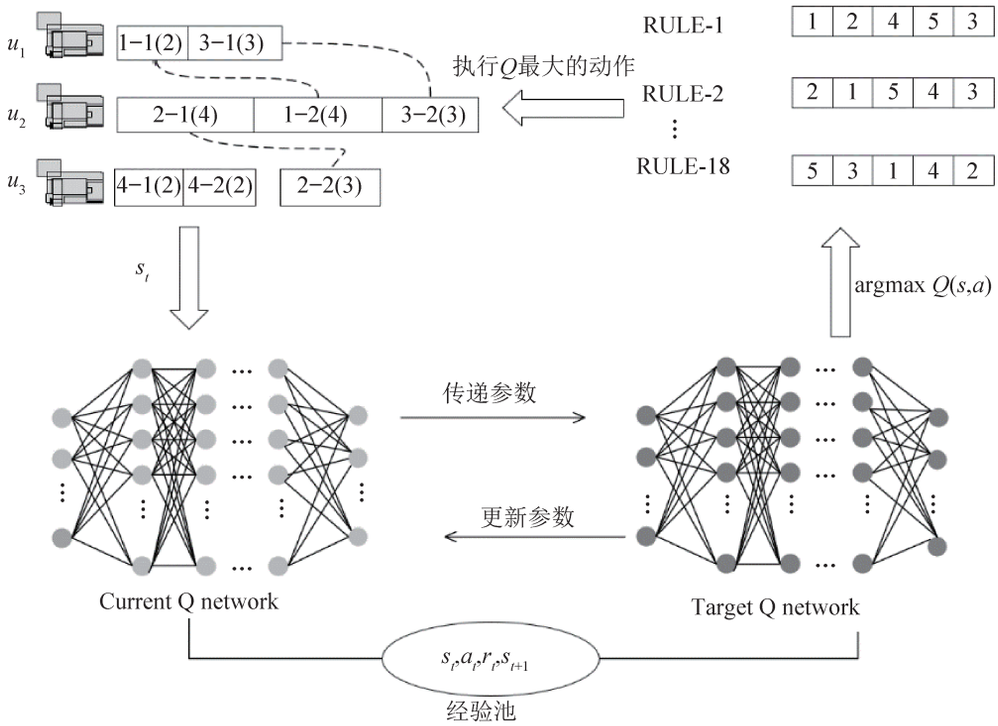



图3 DDQN算法框架图
Fig. 3 DDQN algorithm framework

### 3.3 搜索策略

搜索策略是指智能体的调度决策按照一定的概率进入神经网络进行经验学习，同时也去搜索其他能够获得更高回报的随机行为。*ε-greedy*是较常见的搜索策略，即决策以$\varepsilon$的概率随机选择动作，以1-$\varepsilon$的概率选择$Q$值最大的动作，但该策略在收敛性差，迭代后期还会取选取次优动作。因此本文设置了根据算法迭代次数自适应的搜索策略$\pi^*$，在迭代初期智能体并未有充足的学习经验，随机性大更能充分的探索可行解，而迭代后期的搜索的随机性应大大减少，以更大概率使用最大$Q$值来进行动作选择，使得可行解更快的收敛为

$$\pi^*(s,a)=e_{\mathrm{f}}+\left(e_{\mathrm{s}}-e_{\mathrm{f}}\right)\cdot\exp\left(-\max step/e_{\mathrm{d}}\right) \quad (27)$$

搜索开始概率$e_{\mathrm{s}}$随着最大步长*maxstep*逐步衰减到结束概率$e_{\mathrm{f}}$，其中$e_{\mathrm{d}}$为衰减系数。随机搜索的变化如图4所示，通过实验对比确定参数为$e_{\mathrm{s}}$=1，$e_{\mathrm{f}}$=0.01，$e_{\mathrm{d}}$=100，表示在1 000次的迭代过程中，随机概率由刚开始的完全随机，到迭代后期以0.01的概率随机搜索的逐渐衰减过程。

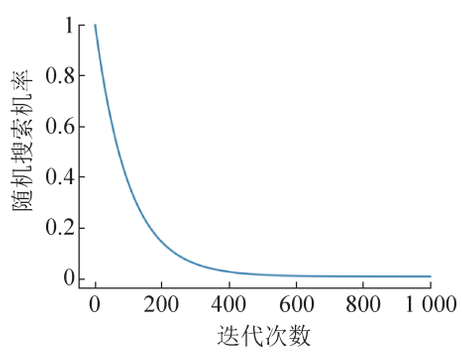



图4 搜索策略示意图
Fig. 4 Search strategy

### 3.4 算法步骤

基于DDQN算法的动态跨单元调度步骤如下所示，调度时刻新工件到达和工件加工完成。




DOI: 10.16182/j.issn1004731x.joss.22-0666

step 1：初始化参数和配置神经网络，包括学习率$\alpha$，折扣因子$\gamma$，采样样本量$Batchsize$，经验池$D$大小，目标Q更新频率等；

step 2：开始一次完整的调度过程，初始化环境状态$s_0$，待加工集合$wait_{\text{job}}$；

step 3：按照式(27)中的搜索策略选择调度动作$\alpha$；

step 4：根据选择的动作和当前的状态，更新待加工集合$wait_{\text{job}}$，删除与已选择的加工工序增加新到达的工件，并计算下一个状态$s_1$；

step 5：计算动作$\alpha$的即时奖励$r_t$，同时将产生的经验数据存储到经验池中，若超过容量阈值，按照先进先出的原则更新经验池的数据；

step 6：当经验池中的经验数据达到开始学习阈值$N$时，从经验池中随机抽取小批量样本，计算当前$Q$值和目标$Q$值；否则转到step 5；

step 7：根据式(25)计算损失函数，更新Q网络参数，学习次数加1；

step 8：当学习次数达到更新频率时，当前$Q$值完全复制到目标Q网络；否则，若此时$wait_{\text{job}}$为空，所有工件调度结束，迭代次数增加1，当总迭代次数小于设置的最大Episode时，转至step 2；若达到最大Episode，输出调度过程中最大完工时间最短的调度方案，算法结束。

# 4 实验与分析

## 4.1 数据说明

为了验证上述DDQN算法在解决动态跨单元调度问题的有效性，本文选取柔性作业车间调度标准测试问题库中的MK01-MK10进行仿真测试。为了模拟在动态环境下的跨单元调度问题，随机生成工件到达时间，并将$M$个加工机器归置到不完全相同的制造单元中。因此算例数据分别为MK01_03，MK02_03，MK03_03，MK04_03，MK05_03，MK06_05，MK07_03，MK08_05，MK09_05，MK10_05。以MK02_03为例，算例规模为10×6×3即为工件数量，机器数和制造单元数。将MK02_03算例中6个机器归置到3个不完全相同的制造单元，1代表制造单元有此类型的加工设备，0相反。设置单位距离的运输速度$v$为40。MK02_03的单元坐标和设备数据如表3所示。

表3 制造单元及其机器归置

Table 3 Manufacturing units and their machines

| 制造单元 | 坐标 | | 机器 | | | |
|---|---|---|---|---|---|---|
| | $X$ | $Y$ | $k_1$ | $k_2$ | … | $k_6$ |
| $u_1$ | 340 | 350 | 1 | 0 | … | 1 |
| $u_2$ | 200 | 300 | 0 | 1 | … | 0 |
| $u_3$ | 720 | 200 | 0 | 0 | … | 0 |

## 4.2 实验参数

本文将提出的跨单元动态调度DDQN算法在PyCharm中编程实现，硬件配置为InterCoreR5-10510UCPU@2.3 GHz，RAM24 GB。经过多次实验探索，确定DDQN算法参数和神经网络参数如表4所示。

表4 实验参数表

Table 4 Experimental parameters

| 算法 | 参数 | 值 |
|---|---|---|
| 神经网络 | 输入层 | 9 |
| | 隐藏层1 | 128 |
| | 隐藏层2 | 128 |
| | 输出层 | 18 |
| | 优化器 | SGD |
| DDQN | Episode | 1000 |
| | 学习率$\alpha$ | 0.001 |
| | 折扣率$\gamma$ | 0.9 |
| | Batch size | 32 |
| | Memory size | 5000 |
| | Target Q更新频率 | 200 |

使用算例MK02_3的数据验算神经网络的训练效果，其训练效果通过训练误差和损失函数来说明。神经网络的拟合效果如图5所示，可以看出神经网络每次从记忆池中选取32个数据进行训练，当前$Q$值与目标$Q$值的数据变化趋势几乎一致。图6展示了loss函数的迭代值，可以看出loss值在训练中逐渐下降并趋于0，在第1 000次调度决策时就完成损失函数的有效训练。



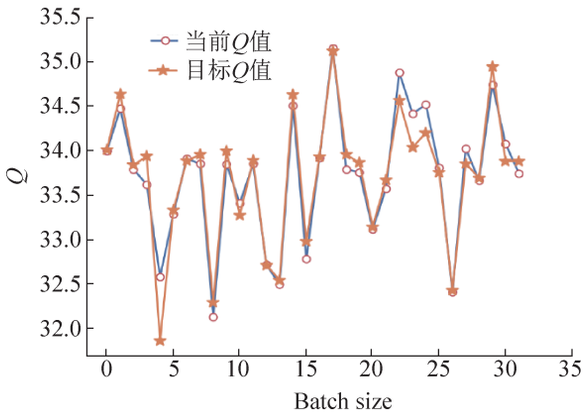



图5 神经网络拟合图

Fig. 5 Neural network fitting

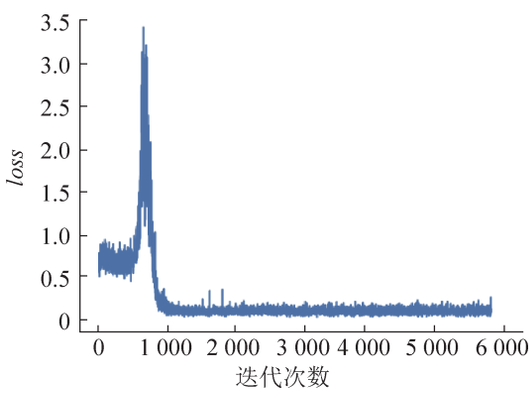



图6 Loss迭代图

Fig. 6 Loss iteration

## 4.3 实验结果及分析

### 4.3.1 算法性能分析

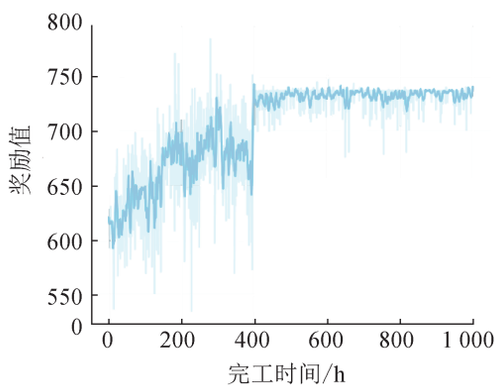
将DDQN算法求解10种规模算例的最大完工时间与18种复合规则的最优解进行比较，DDQN算法求解的最大完工时间均优于单一调度规则。结果对比如表5所示。

以MK05-03测试算例说明采用DDQN算法进行调度决策的过程，图7(a)展示的是该算例在1 000次训练下的总奖励迭代图，浅色部分是程序的原始数据，为了更清晰的展示数据的迭代趋势，采用Savitzky-Golay滤波器对数据进行平滑处理。可以看出忽略个别极值的影响，DDQN算法生成调度方案的总奖励值呈明显上升趋势，在第400次迭代后变化趋于平缓，这是由于最初智能体的决策随机性较高，后期随着经验数据增加学习效果越稳定。图7(b)为最大完工时间的迭代图，可以看出目标函数呈明显下降趋势，说明奖励函数的设置能够引导智能体对最大完工时间的优化。

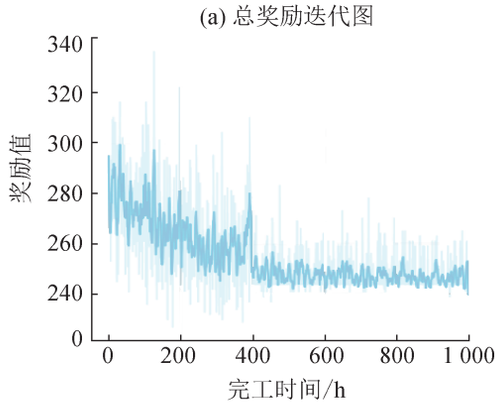
表5 不同算例的实验结果表

Table 5 Experimental results of different examples

| 算例 | DDQN算法 | 复合规则最优解 | 规则编号 |
|---|---|---|---|
| MK01_03 | 67 | 71 | 2 |
| MK02_03 | 91 | 106 | 18 |
| MK03_03 | 342 | 354 | 7 |
| MK04_03 | 115 | 125 | 4 |
| MK05_03 | 230 | 244 | 2 |
| MK06_05 | 217 | 236 | 13 |
| MK07_03 | 290 | 312 | 10 |
| MK08_05 | 579 | 660 | 17 |
| MK09_05 | 521 | 538 | 11 |
| MK10_05 | 417 | 427 | 5 |




(a) 总奖励迭代图




(b) 完工时间迭代图

图7 MK05_03算例迭代图

Fig. 7 MK05_03 example iteration

综上说明，调度智能通过学习经验可以进行较优的调度决策，也验证了本文提出的DDQN算法的有效性。该算例生成最优解的甘特图如图8所示，智能体共执行了106次调度决策，依次执行的复合调度规则的编号为{1, 12, 1, 7⋯1, 5, 8}。




DOI: 10.16182/j.issn1004731x.joss.22-0666

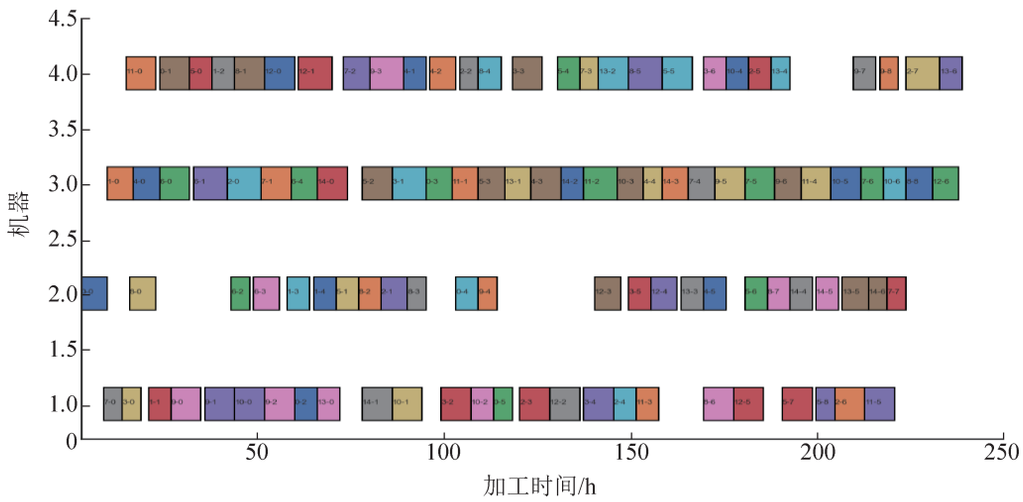



图8 MK05_03最优解甘特图
Fig. 8 MK05_03 Gantt chart of optimal solution

为了检验本文提出的DDQN算法求解优越性，将*ε-greedy*搜索策略和本文所提出的搜索策略$\pi^*$在DDQN算法和DQN算法的应用上进行独立对比分析，设置$\varepsilon=0.2$。同时设计了基于模拟退火算法(simulated annealing，SA)的对比实验，设置模拟退火算法的解为每一次完整调度的规则向量。10种算例5组实验的结果对比如表6所示。

由表6可知，基于搜索策略$\pi^*$的DDQN算法和DQN算法求解的完工时间均优于基于*ε-greedy*策略的算法求解结果，并且2种策略的DDQN算法的完工时间均优于DQN算法和模拟退火算法。

以MK08_05算例的求解结果为例，将基于自适应搜索策略$\pi^*$的DDQN和DQN算法求解迭代中的两组1 000个对比数据通过小提琴图进进行分析，小提琴图除了能够展现数据分布之外，以更清晰的看出数据分布密度。图9可以看出DDQN求解的总奖励值数据分布的四分位数、中位数相较于DQN算法都更高，而完工时间值分布的四分位数、中位数相较于DQN都更低，并且在较优解附近分布密度较大。

### 4.3.2 动态因素下的实例分析

DDQN算法应用在调度领域的一大优势是能够灵活地处理不同情况下环境干扰实现自适应调度。以某制造企业的生产实例作为调度案例进行实验分析，得到10×10×5的大规模算例。实验通过随机更改工序的加工时间和制造单元的坐分别用于模拟生产实际中的加工拖期和运输时间变化的动态因素。

表6 不同算法策略求解表
Table 6 Solution table of different algorithm strategies

| 算例 | DDQN ($\pi^*$) | DDQN (*ε-greedy*) | DQN ($\pi^*$) | DQN (*ε-greedy*) | SA |
|---|---|---|---|---|---|
| MK01_03 | **67** | 75 | 79 | 83 | 95 |
| MK02_03 | **91** | 105 | 105 | 125 | 132 |
| MK03_03 | **342** | 350 | 366 | 379 | 367 |
| MK04_03 | **115** | 139 | 153 | 159 | 159 |
| MK05_03 | **230** | 252 | 244 | 257 | 292 |
| MK06_05 | **217** | 247 | 253 | 272 | 267 |
| MK07_03 | **290** | 322 | 329 | 347 | 335 |
| MK08_05 | **579** | 636 | 631 | 655 | 701 |
| MK09_05 | **521** | 530 | 542 | 562 | 553 |
| MK10_05 | **417** | 424 | 429 | 437 | 521 |

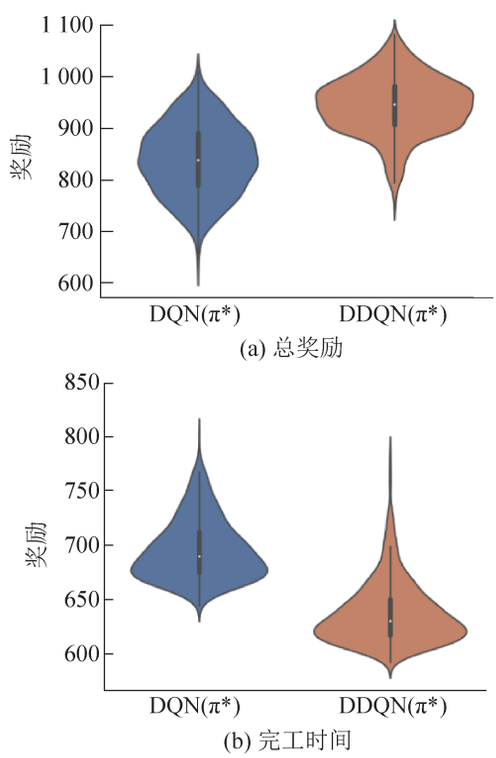
首先将DDQN算法采用表4参数的求解实际算例，运行结果如图10所示，可以看出总奖励和完工时间的迭代呈相反的趋势，说明DDQN算法在求解实际算例中的有效性。其次将含有最优策略的神经网络用于求解更改后的实际算例，其运行结果如图11所示，可以看出算法收敛速度明显



提升，完工时间初始值为1 178，这是当前模型在环境变化时给出的即时调度方案。随着训练过程的继续，模型经过180次迭代收敛并且最大完工时间稳定在1 200左右。DDQN算法在求解调度问题上的价值不仅是输出调度方案用于指导实际调度，并且其可以在不断变化的环境中学习到最优的调度策略，因此能够响应动态的生产环境，在加工时间延期、运输时间变化的情况下快速决策生成可行的调度方案。

接着对算法执行时间进行对比分析，18种复合调度规则的平均执行时间为7 s，然而，由于单一规则不能满足每种情况下的调度问题，因此无法获得良好的结果。DDQN算法可以在3 s内做出单一规则更好的决策，表明模型能够立即适应环境的变化，从而实现自适应调度。综合求解结果和算法时间的分析可以认为训练后的模型具有一定的泛化能力，能够处理跨单元调度过程中的不确定因素。

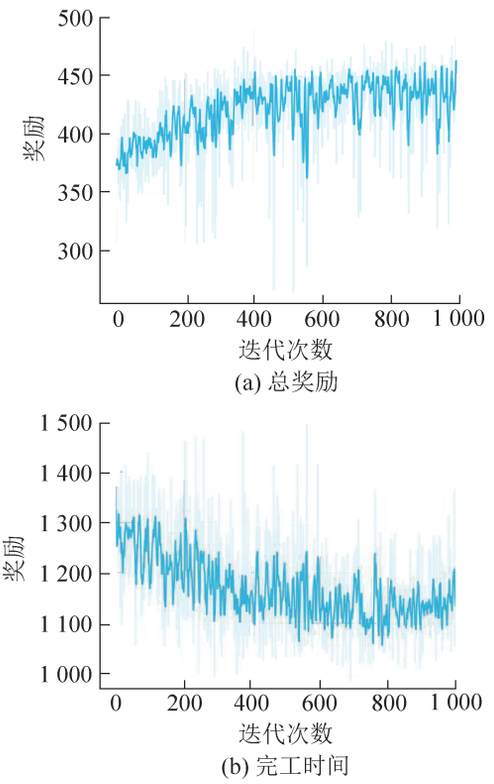



(a) 总奖励

(b) 完工时间

图9 两种算法求解MK08_05小提琴图

Fig. 9 Two algorithms for solving MK08_05 violin

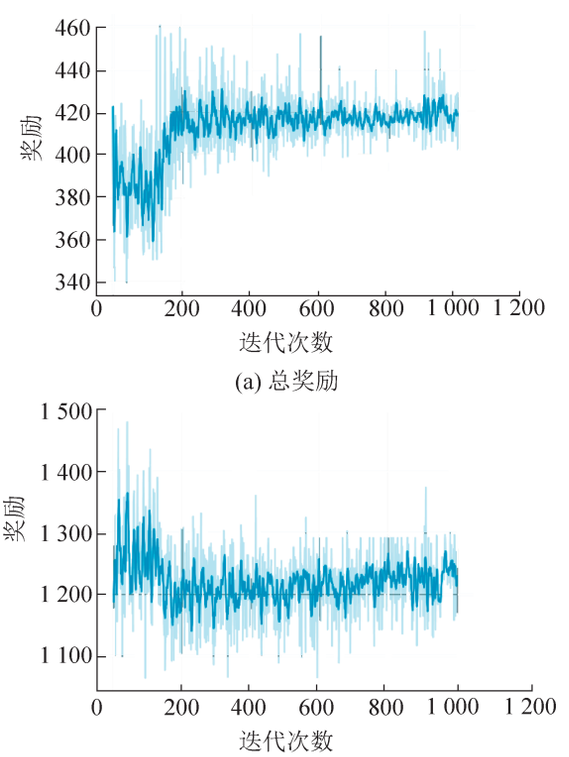



(a) 总奖励

(b) 完工时间

图10 DDQN求解实例迭代图

Fig. 10 Iteration of DDQN solution example




(a) 总奖励

(b) 完工时间

图11 含有调度策略的神经网络求解迭代图

Fig. 11 Iteration of neural network solution with scheduling strategy




DOI: 10.16182/j.issn1004731x.joss.22-0666

## 5 结论

本文提出了一种基于深度强化学习的调度方法，以解决工件随加工任务动态到达的分布式单元调度问题。首先根据跨单元调度问题特点定义了以工件、机器和加工单元为主体的状态空间，使用网络度值来描述制造单元的状态，扩充了智能体的训练数据。其次将跨单元调度分解为单元层和机器层的调度，设计了由工件、单元和机器组成的复合调度规则，使得调度方案能够平衡各个单元和机器的加工负荷。在仿真算例的验证下，DDQN 算法的求解效果要优于单一的调度规则和进化算法，并且能够在动态的生产环境中较快输出调度结果。同时通过实验发现算法在解决大规模问题的收敛性低于解决小规模问题，因为大规模算例的状态空间巨大，需要更优化的网络结构和迭代次数来减少训练误差。未来可以进一步探索其他复杂网络特征与调度问题的结合，从而更全面地刻画调度系统的状态变化。

## 参考文献：

[1] Tang Jiafu, Zeng Chengkuan, Pan Zhendong. Auction-based Cooperation Mechanism to Parts Scheduling for Flexible Job Shop with Inter-cells[J]. Applied Soft Computing, 2016, 49: 590-602.

[2] Li Dongni, Wang Yan, Xiao Guangxue, et al. Dynamic Parts Scheduling in Multiple Job Shop Cells Considering Intercell Moves and Flexible Routes[J]. Computers & Operations Research, 2013, 40(5): 1207-1223.

[3] Huang Z, Yang J J. A New Model for Optimization of Cell Scheduling Considering Inter-cell Movement[J]. International Journal of Simulation Modeling, 2022, 21 (1): 136-147.

[4] Deliktas D, Torkul O, Ustun O. A Flexible Job Shop Cell Scheduling with Sequence-dependent Family Setup Times and Intercellular Transportation Times Using Conic Scalarization Method[J]. International Transactions in Operational Research, 2019, 26(6): 2410-2431.

[5] Liu Chunfeng, Wang Jufeng, Leung J Y T, et al. Solving Cell Formation and Task Scheduling in Cellular Manufacturing System by Discrete Bacteria Foraging Algorithm[J]. International Journal of Production Research, 2016, 54(3): 923-944.

[6] 曾程宽, 刘士新. 求解存在运输空间约束多单元协作调度问题的拍卖算法[J]. 控制与决策, 2019, 34(4): 689-698.
Zeng Chengkuan, Liu Shixin. Auction-based Cooperation Mechanism for Cell Part Scheduling with Transportation Capacity Constraint[J]. Control and Decision, 2019, 34 (4): 689-698.

[7] 连永伟, 董钊睿, 刘琼. 跨单元调度及其车辆路径集成优化[J]. 中国机械工程, 2022, 33(6): 747-755.
Lian Yongwei, Dong Zhaorui, Liu Qiong. Integrated Optimization of Intercell Scheduling and Vehicle Routing [J]. China Mechanical Engineering, 2022, 33(6): 747-755.

[8] Zhao Meng, Li Xinyu, Gao Liang, et al. An Improved Q-learning Based Rescheduling Method for Flexible Job-shops with Machine Failures[C]//2019 IEEE 15th International Conference on Automation Science and Engineering (CASE). Piscataway, NJ, USA: IEEE, 2019: 331-337.

[9] Wang Yufang. Adaptive Job Shop Scheduling Strategy Based on Weighted Q-learning Algorithm[J]. Journal of Intelligent Manufacturing, 2020, 31(2): 417-432.

[10] Luo Shu. Dynamic Scheduling for Flexible Job Shop with New Job Insertions by Deep Reinforcement Learning[J]. Applied Soft Computing, 2020, 91: 106208.

[11] Zhao Yejian, Wang Yanhong, Tan Yuanyuan, et al. Dynamic Jobshop Scheduling Algorithm Based on Deep Q Network[J]. IEEE Access, 2021, 9: 122995-123011.

[12] Lang S, Behrendt F, Lanzerath N, et al. Integration of Deep Reinforcement Learning and Discrete-event Simulation for Real-time Scheduling of a Flexible Job Shop Production[C]//2020 Winter Simulation Conference (WSC). Piscataway, NJ, USA: IEEE, 2020: 3057-3068.

[13] Zhou Tong, Tang Dunbing, Zhu Haihua, et al. Reinforcement Learning with Composite Rewards for Production Scheduling in a Smart Factory[J]. IEEE Access, 2021, 9: 752-766.

[14] Palombarini J A, Ernesto C Martínez. Closed-loop Rescheduling Using Deep Reinforcement Learning[J]. IFAC-PapersOnLine, 2019, 52(1): 231-236.

[15] Kardos C, Laflamme C, Gallina V, et al. Dynamic Scheduling in a Job-shop Production System with Reinforcement Learning[J]. Procedia CIRP, 2021, 97: 104-109.

[16] Holthaus O, Rajendran C. Efficient Dispatching Rules for Scheduling in a Job Shop[J]. International Journal of Production Economics, 1997, 48(1): 87-105.

[17] Wei Yingzi, Zhao Mingyang. Composite Rules Selection Using Reinforcement Learning for Dynamic Job-shop



Scheduling[C]//IEEE Conference on Robotics, Automation and Mechatronics. Piscataway, NJ, USA: IEEE, 2004: 1083-1088.

[18] Wang Sen, Zhang Peng, Qin Wei, et al. Composite Dispatching Rule Design for Photolithography Area Scheduling in Wafer Manufacturing System with Multiple Objectives[J]. Applied Mechanics and Materials, 2013, 252: 418-421.

[19] Han Baoan, Yang Jianjun. Research on Adaptive Job Shop Scheduling Problems Based on Dueling Double DQN[J]. IEEE Access, 2020, 8: 186474-186495.

[20] 邹萌邦. 基于复杂网络特征的神经网络调度器求解车间调度问题研究[D]. 武汉: 华中科技大学, 2019.
Zou Mengbang. Research on Complex Network Features Based Neural Network Scheduler for Job Shop Scheduling Problem[D]. Wuhan: Huazhong University of Science and Technology, 2019.

[21] 肖鹏飞, 张超勇, 孟磊磊, 等. 基于深度强化学习的非置换流水车间调度问题[J]. 计算机集成制造系统, 2021, 27(1): 192-205.
Xiao Pengfei, Zhang Chaoyong, Meng Leilei, et al. Non-permutation Flow Shop Scheduling Problem Based on Deep Reinforcement Learning[J]. Computer Integrated Manufacturing Systems, 2021, 27(1): 192-205.




DOI: 10.16182/j.issn1004731x.joss.22-0666