doi:10.3969/j.issn.1673-9833.2024.04.006

# 基于图卷积网络求解开放车间调度问题的方法

赵昊鑫[1,2]，万烂军[1,2]，崔雪艳[1,2]，李长云[1,2]

（1. 湖南工业大学 计算机学院，湖南 株洲 412007；2. 湖南工业大学 智能信息感知及处理技术湖南省重点实验室，湖南 株洲 412007）

**摘 要**：传统的元启发式算法难以有效求解大规模开放车间调度问题（OSSP），为此提出了一种基于图卷积网络 GCN 求解 OSSP 的方法。首先，设计了基于 GCN 的开放车间调度模型，将 OSSP 的工序节点特征嵌入图中并对其进行多层卷积操作，有效获取了工序节点之间复杂的依赖关系。然后，为了提高求解大规模 OSSP 的效率和质量，提出了一种基于 GCN 的开放车间调度算法。实验结果表明，该方法能有效求解不同规模的 OSSP 实例，与元启发式算法相比，在求解大规模 OSSP 实例时该方法表现出更优秀的求解质量和效率。

**关键词**：开放车间调度；图卷积网络；元启发式算法

**中图分类号**：TP18 **文献标志码**：A **文章编号**：1673-9833(2024)04-0034-06

**引文格式**：赵昊鑫，万烂军，崔雪艳，等 . 基于图卷积网络求解开放车间调度问题的方法 [J]. 湖南工业大学学报，2024，38(4)：34-39.

# A Method for Solving Open Workshop Scheduling Problem Based on Graph Convolutional Network

ZHAO Haoxin[1,2]，WAN Lanjun[1,2]，CUI Xueyan[1,2]，LI Changyun[1,2]

（1. College of Computer Science，Hunan University of Technology，Zhuzhou Hunan 412007，China；
2. Hunan Key Laboratory of Intelligent Information Perception and Processing Technology，Hunan University of Technology，Zhuzhou Hunan 412007，China）

**Abstract**：Due to the lower efficiency found in traditional metaheuristic algorithms for a solution of a large-scale open workshop scheduling problems (OSSP), a method based on graph convolution network (GCN) has thus been proposed to solve OSSP. Firstly, an open workshop scheduling model based on GCN has been designed, which incorporates the process node features of OSSP into the graph, to be followed by a multi-layer convolution operation on it, thus effectively obtaining the complex dependency relationships between process nodes. Next, in view of an improvement of efficiency and quality of solving large-scale OSSP, an open workshop scheduling algorithm has been proposed based on GCN. The experimental results show that this method can effectively solve OSSP instances of different scales. Compared with metaheuristic algorithms, the proposed method is characterized with a better solution quality and higher efficiency in solving large-scale OSSP instances.

**Keywords**：open shop scheduling；graph convolutional network；meta-heuristic algorithm

**收稿日期**：2023-06-21

**基金项目**：国家自然科学基金资助项目（61702177）；湖南省教育厅优秀青年基金资助项目（21B0547）；湖南省自然科学基金资助项目（2023JJ30217）；湖南省教育厅科研基金资助重点项目（21A0356）

**作者简介**：赵昊鑫，男，湖南工业大学硕士生，主要研究方向为工业大数据分析，E-mail：zhaohaoxin@stu.hut.edu.cn

**通信作者**：万烂军，男，湖南工业大学副教授，博士，主要研究方向为工业大数据与并行计算，E-mail：wanlanjun@hut.edu.cn

## 1　背景知识

OSSP（open shop scheduling problem）在电子制造、汽车制造、航空航天等制造业领域具有广泛的应用场景，涉及对多个机台上的多道工序进行有效调度，以最小化生产时间或最大化生产效率。其解决方案可以帮助企业优化生产计划，提高生产效率，减少生产成本，降低制造周期等 [1−2]。

元启发式算法是一种通用的启发式算法框架，它结合了不同的启发式算法，通过模拟自然和人类智慧来实现最优解的求解，高效地解决了各种优化问题。目前用于求解 OSSP 的元启发式算法主要包括遗传算法（genetic algorithm，GA）[3−6]、蚁群优化算法（ant colony optimization algorithm，ACO）[7−9] 和粒子群优化算法（particle swarm optimization，PSO）[10−12] 等。Wang W. J. 等 [3] 提出了一种基于可变邻域搜索的混合遗传算法求解方法，并根据优化目标设计了 3 种不同的解码方法。通过仿真实验，证明了该方法在解决调度问题方面的可行性与有效性。L. R. Abreu 等 [5] 提出了一种新的带有迭代贪婪局部搜索过程的有偏随机密钥 GA，用于有容量车辆路由的 OSSP。H. A. A. Rahmani 等 [6] 考虑了交叉和变异算子对求解 OSSP 的影响，提出了一种用于求解 OSSP 的改进 GA。M. Kurdi[8] 通过引入一种新的探索性启发式信息方法，并结合 3 个探索性特征，提出了一种高效求解 OSSP 的 ACO，提高了蚁群的探索能力。C. Blum[9] 提出了一种将 ACO 的求解构造机制与波束搜索结合的混合 ACO，该算法增强了 ACO 的稳定性，提高了求解 OSSP 的质量和效率。V. Kachitvichyanukul 等 [11] 采用两级 PSO 求解 OSSP，上层 PSO 微调下层 PSO 的参数，下层 PSO 使用给定参数生成解，该方法在 OSSP 基准实例上表现良好。Sha D. Y. 等 [12] 提出了一种适用于求解 OSSP 的 PSO，通过修改粒子位置、粒子运动和粒子速度的表示来提高适应性，并引入了新的解码算子来获得更好的计算结果。

GA、ACO 和 PSO 等元启发式算法在求解 OSSP 中具有一定的优势。然而，在实际应用中，随着机台数量、工件数量和工序数量的增加，开放车间调度问题的规模和复杂度呈指数级别地增长，元启发式算法 [13] 面临着多种局限性，如面临时间复杂度高、易陷入局部最优解、参数调节困难等问题。

传统求解 OSSP 的方法往往需要大量的搜索空间，不断扩大的开放车间调度问题规模对它们的求解质量和求解效率会产生较大的影响。为了提高大规模 OSSP 的求解质量和求解效率，本文提出一种通过基于图卷积网络的链路预测求解大规模 OSSP 的方法，并采用不同规模的 OSSP 基准实例和随机生成的 OSSP 实例，开展实验来验证提出方法的有效性。

## 2　开放车间调度问题

OSSP 是一个典型的 NP-hard 问题，定义如下：在一个车间中，有 $m$ 个机台和 $n$ 个工件，每个工件包含 $m$ 道工序，每道工序都必须在预定的机台上进行加工处理。其中，$P_{ij}$、$S_{ij}$、$E_{ij}$ 分别表示工序 $O_{ij}$ 的加工时间、开始加工时间和完工时间；$M_j$ 和 $J_j$ 表示第 $i$ 个机台。$P_{ijk}$、$E_{ijk}$ 分别表示工序 $O_{ij}$ 在机台 $k$ 上的加工时间和完工时间。$C_m$、$C_{\max}$ 分别表示机台 $M_m$ 的完工时间和调度实例的最大完工时间。相关符号定义说明如表 1 所示。

**表 1　相关符号定义说明**

Table 1　Definition and explanation of relevant symbols

| 符号 | 定义说明 | 符号 | 定义说明 |
|---|---|---|---|
| $m$ | 机台的数量 | $S_{ij}$ | 工序 $O_{ij}$ 的开始加工时间 |
| $n$ | 工件的数量 | $E_{ij}$ | 工序 $O_{ij}$ 的结束加工时间 |
| $M_i/J_i$ | 第 $i$ 个机台，其中 $1 \leqslant i \leqslant n$ | $E_{ijk}$ | 工序 $O_{ij}$ 在机台 $k$ 上的结束加工时间 |
| $O_{ij}$ | 工件 $J_i$ 的第 $j$ 道工序 | $S_i$ | 工件 $J_i$ 的开始加工时间 |
| $P_{ij}$ | 工序 $O_{ij}$ 的加工时间 | $E_i$ | 工件 $J_i$ 的结束加工时间 |
| $P_{ijk}$ | 工序 $O_{ij}$ 在机台 $k$ 上的加工时间 | $C_m$ | 机台 $M_m$ 的完工时间 |
| | | $C_{\max}$ | 调度实例的最大完工时间 |

对于一个 OSSP 实例，一个可行的调度方案里面应该包括两个序列，工序序列和机台序列。工序序列表示的是每个工件工序的先后加工顺序，机台序列表示的是每个工件在每个机台上的加工顺序。OSSP 的约束为一个机台在同一时刻只能加工一个工件，一个工件的工序在同一时刻只能在一个机台上进行加工，且允许机台空闲。具体约束如下：式（1）表示在某一时刻 $O_{ij}$ 在 $J_k$ 加工时，$W_{ijk}$=1；式（2）为每个作业的各道工序在同一时刻只能在一个机台上进行加工；暂时不考虑抢占式调度问题，因此式（3）为所有操作一旦开始执行，将不被打断地执行，直到工序执行结束、工件离开机台；式（4）确保如果工件 $J_i$ 应在工件 $J_j$ 之前在机台 $M_k$ 上加工，$Y_{ijk}$=1；式（5）为每个机台在一段时间内只能对一道工序进行加工，即工序 $O_{ihk}$ 必须在工序 $O_{jgk}$ 开始加工之前完成，其中 $C_{ihk}$ 和 $C_{jgk}$ 分别是工序 $O_{ihk}$ 和工序 $O_{jgk}$ 的结束加工时间，$t_{jgk}$ 是工序 $O_{jgk}$ 的加工时间，$M$ 为一个较大的正数；式（6）~（9）确定变量的范围。式（10）表示的最大完工时间是所有工件完工时间的最大值。本文的目标函数为式（11），即在不规定每个工件的加工顺序的情况下，安排调度方案使得最大完工时间最小化 [14]。

$$W_{ijk}=\begin{cases}1, \text{ if } O_{ij} \text{ is processed on } M_k;\\ 0, \text{ otherwise}。\end{cases} \quad (1)$$

$$\sum_{k=1}^{m} W_{ijk}=1,\ \forall i,\ j。 \quad (2)$$

$$S_{ij}+W_{ijk}\times P_{ij}=E_{ij},\ \forall i,\ j,\ k。 \quad (3)$$

$$Y_{ijk}=\begin{cases}1, \text{ if } J_i \text{ is processed on } M_k \text{ before } J_j;\\ 0, \text{ otherwise}。\end{cases} \quad (4)$$

$$C_{jgk}-C_{ihk}+(1-Y_{ijk})\times M\geqslant t_{jgk},\ \forall i,\ j,\ k,\ i\neq j。 \quad (5)$$

$$C_{\max}\geqslant C_{ij},\ \forall i<n,\forall j<m。 \quad (6)$$

$$C_{\max},\ C_{ij}\geqslant 0,\ \forall i<n,\forall j<m。 \quad (7)$$

$$W_{ijk}\in\{0,\ 1\},\ \forall i<n,\ \forall j,\ k<m, \quad (8)$$

$$Y_{ijk}\in\{0,1\},\forall i<n,\forall j,\ k<m, \quad (9)$$

$$C_{\max}=\max\{C_1,\ C_2,\ \cdots,\ C_m\}。 \quad (10)$$

$$\text{Minimize } C_{\max}。 \quad (11)$$

## 3 求解 OSSP 方法设计与实现

### 3.1 基于 GCN 的开放车间调度模型

采用析取图表达 OSSP 中工序之间的关系和机台信息，以便从图的角度更加清晰地描述 OSSP，并对工序的节点特征进行设计，从而能高效利用节点特征信息来求解 OSSP，工序的节点特征如图 1 所示。此外，为充分利用图结构中工序节点之间的关系信息，通过嵌入 OSSP 中的工序节点特征的方式设计了基于 GCN 的开放车间调度模型。

| | $M_1$ | $M_2$ | $M_3$ | … | $M_{m-1}$ | $M_m$ | 时间归一化 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | … | 0 | 0 | $T_0$ |
| 1 | 0 | 0 | 0 | … | 0 | 1 | $T_1$ |
| 2 | 0 | 0 | 0 | … | 1 | 0 | $T_2$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋱ | ⋮ | ⋮ | ⋮ |
| $m\times n-2$ | 1 | 0 | 0 | … | 0 | 0 | $T_{m\times n-2}$ |
| $m\times n-1$ | 0 | 0 | 1 | … | 0 | 0 | $T_{m\times n-1}$ |

**图 1 工序节点特征设计图**

Fig. 1 Operation node feature design diagram

传统的深度学习模型，如长短期记忆网络和卷积神经网络，在欧式空间中表现优异，但是却无法直接被应用在非欧式数据上，因此本文采用图论中的抽象概念“图”来表征非欧几里得空间的结构化数据，通过 GCN 来提取图的拓扑结构，挖掘图结构数据中的深层次信息。

每个节点在图中都不断受到邻居节点和更远节点的影响，以至于它们的状态不断变化，直到达到最终的平衡状态。这种影响的强度与邻居之间的亲密程度成正比。

GCN 是谱图卷积的局部一阶近似。本文设计的模型规模会随图中边的数量增长而线性增长，并学习编码局部图结构和节点特征的隐藏层表示。分层传播规则的多层图卷积网络的前向传播，通过式（12）计算。

$$\boldsymbol{H}^{(l+1)}=\text{sigmoid}\left(\tilde{\boldsymbol{D}}_{ii}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}_{ij}\tilde{\boldsymbol{D}}_{ii}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right)。 \quad (12)$$

图的自环邻接矩阵 $\boldsymbol{B}_{ij}$ 通过式（13）计算。

$$\boldsymbol{B}_{ij}=\boldsymbol{A}_{ij}+\boldsymbol{I}_N。 \quad (13)$$

$\boldsymbol{B}_{ij}$ 的度矩阵 $\boldsymbol{D}_{ii}$ 通过公式（14）计算。

$$\boldsymbol{D}_{ii}=\sum\nolimits_j \boldsymbol{B}_{ij}。 \quad (14)$$

式（12）~（14）中：$\boldsymbol{W}^{(l)}$ 为可训练的权重矩阵；sigmoid 为激活函数；$\boldsymbol{H}^{(l+1)}$ 为第 $l$ 层的激活矩阵；$\boldsymbol{A}_{ij}$ 为工序节点 $O_{ij}$ 的特征矩阵；$\boldsymbol{I}_N$ 为秩是 $N$ 的单位矩阵；$\boldsymbol{H}^{(0)}=\boldsymbol{X}$ 为第一层的输入矩阵。

本文使用具有以下分层传播规则的多层 GCN，GCN 模型迭代更新主要基于图中每个节点的邻居信息，采用 7 层图卷积网络和 1 层 sigmoid 函数，根据每个工序节点本身的特征信息和邻居信息，对大规模 OSSP 中的每个工序节点进行节点嵌入，得到多层 embedding 之后的每个工序节点的特征信息。

图 2 展示了基于 GCN 的开放车间调度模型的网络结构示意图，将大规模 OSSP 的实例作为一个完整的图，将其输入设计好的网络模型结构中，输入网络模型结构中的信息包括各个工序节点的输入特征以及网络结构图。图卷积层可以结合邻居工序节点与自身节点的信息，更新自身工序节点的特征信息，每一层图卷积层将当前每个工序节点的特征信息和网络结构图依次传入下层，得到更新后的工序节点的特征信息。

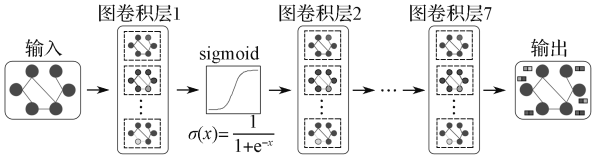


**图 2 基于 GCN 的开放车间调度模型的网络结构图**

Fig. 2 Network structure diagram of the GCN-based open workshop scheduling model

### 3.2 基于 GCN 的开放车间调度算法

为了高效地求解大规模 OSSP，设计了一种基于 GCN 的开放车间调度算法。该算法可以根据预测结果生成加工序列，以尽可能满足预测出的工序之间的交互关系。同时，这个算法还能通过调整加工序列的顺序来优化解决方案，最终目标是使所有工件的最大完工时间最小化。基于 GCN 的开放车间调度算法的

具体步骤如下。

第一步：从 $\boldsymbol{X}$ 的第一行开始遍历，查找在同一机台上进行加工的工序，得到过滤之后的特征矩阵 $Fillter(\boldsymbol{X})$，并对所有工序的结束时间 $T_{\text{te}}$ 赋初值 −1。

第二步：构造每个机台的第一个工序节点。

第三步：查找当前待选工序节点所属工件的其他所有已经确定结束时间的工序节点中的最大结束时间 $T_{\text{me}}$，获取当前机台的上一道工序的结束时间 $T_{\text{ce}}$。

第四步：通过式（15）计算 $T_{\text{wait}}$。

$$T_{\text{wait}}=\begin{cases}0, & \text{if } T_{\text{me}}<T_{\text{ce}};\\ T_{\text{me}}-T_{\text{ce}}, & \text{otherwise}。\end{cases} \tag{15}$$

第五步：通过式（16）计算每个作业的第一个工序节点的得分。

$$W_{\text{score}}=F_x\times W_{\text{feature}}+T_{\text{wait}}+W_{\text{wait}}。 \tag{16}$$

式中：$F_x$ 为工序节点 $x$ 特征归一化后的值；$W_{\text{feature}}$ 为工序节点特征分配的系数；$W_{\text{wait}}$ 为等待时间分配系数。

第六步：经过排序选出每个机台上的第一个工序节点，并将结果输入调度结果 $\boldsymbol{R}$ 中。

第七步：构造每个机台剩下的工序节点。首先，计算工序节点的概率。然后，确定该节点是否是机台上的最后一道工序。如果是，将最后一点作为机台调度的最后一道工序；如果不是，则最终分数由式（17）计算得出。

$$W_{\text{score}}=S_x\times W_{\text{feature}}+T_{\text{wait}}+W_{\text{wait}}。 \tag{17}$$

式中 $S_x$ 为剩余工序节点 $x$ 特征归一化后的值。

第八步：取出每个机台最终调度结果的最后一道工序的结束时间，最大的结束时间即为整个调度实例的 $C_{\max}$。根据调度结果 $\boldsymbol{R}$ 画甘特图。

# 4　实验

## 4.1　实验设置

为评估提出的基于 GCN 求解大规模 OSSP 方法的有效性，针对来源于 E. Taillard[15] 数据集中 OSSP 基准实例和随机生成的 OSSP 实例开展了大量实验。为保证实验结果的可靠性，对每个实例重复进行 10 次测试，每次测试都记录该实例的最大完工时间。对每个实例完成 10 次测试后，将最小的最大完工时间作为最佳解（BS）。

OSSP 基准实例具有如下 3 种不同的规模：$10\times10$、$15\times15$ 和 $20\times20$。随机生成的 OSSP 实例具有如下 10 种不同的规模：$30\times20$、$40\times20$、$50\times20$、$60\times20$、$70\times20$、$80\times20$、$90\times20$、$100\times20$、$150\times20$ 和 $200\times20$。在随机生成的 OSSP 实例中，所有实例的工序加工时长是随机生成的，取值为 1~50 的随机整数。

在实验中，提出的用于求解 OSSP 的 GCN 模型由 7 个图卷积层和 1 个 sigmoid 层组成。图卷积层 1~7 中的隐藏层数量分别为 19, 17, 13, 11, 7, 5, 3。最终输出的每个工序节点的特征数目为两个。在 GCN 模型中采用具有固定学习率 $2\times10^{-3}$ 的 Adam 优化器。

实验平台的硬件配置如下：八核的 Intel Core i7-9700K CPU @3.60 GHz 以及 64 GB 内存。实验平台的软件配置如下：CentOS 8.1、Python 3.8.3、Pytorch 1.6.0 和 PyG 2.3。

## 4.2　与元启发式算法比较

为验证提出的基于 GCN 求解大规模 OSSP 方法的有效性，针对每个 OSSP 基准实例，分别采用 3 种不同的元启发式算法（也即 GA[6]、ACO[9] 和 PSO[12]）以及提出的 GCN 方法进行求解。这 3 种元启发式算法的关键参数设置如下：

1）GA。种群大小为 100，交叉概率为 0.8，变异概率为 0.05，最大迭代次数为 200。

2）ACO。蚂蚁数量为 100，信息素挥发因子为 0.05，信息素重要度因子为 1.0，启发式因子为 1.0，最大迭代次数为 200。

3）PSO。粒子数量为 100，学习因子和社会因子为 2.0，惯性权重为 0.8，最大迭代次数为 200。

在 GA 中，种群是指所有可行解的集合，即搜索空间中的一个子集。在 OSSP 中，每个调度方案代表了不同的工件在不同的机台上的调度顺序。一个种群由多个调度方案（个体）组成，每个调度方案是一组任务的调度序列。个体是种群中的一个单元，它代表了一个具体的调度方案。在 OSSP 中，每个个体由任务序列组成，每个任务序列表示了工件在不同机台上的执行顺序。

表 2 展示了采用 GA、ACO、PSO 和 GCN 分别求解 Taillard 数据集中 $10\times10$、$15\times15$ 和 $20\times20$ 这 3 种不同规模 OSSP 基准实例的最佳 Makespan。如果采用 GA、ACO、PSO 和 GCN 求解出的 Makespan 与目前已知最佳的求解方案（best known solutions，BKS）相等，则用粗体表示。

如表 2 所示，在大多数实例中，采用 GA、ACO 和 PSO3 种元启发式算法都能获得与 BKS 相等的结果，而采用提出的 GCN 方法在所有实例上均能获得与 BKS 相等的结果。这可能是因为元启发式算法在求解优化问题时通常采用随机化策略，通过不断搜索寻找最优解，但在某些情况下可能会陷入局部最优解，无法得到全局最优解。而基于 GCN 方法则是通过图神经网络对求解 OSSP 的任务进行建模，并利用图卷积层进行特征提取和信息传递，从而得到全局最

优解。

表 2 OSSP 基准实例的求解结果

Table 2 Solution results for OSSP benchmark instances

| 实例名称 | BKS | GA | ACO | PSO | GCN |
|---|---|---|---|---|---|
| tai_10×10_1 | **637** | **637** | **637** | **637** | **637** |
| tai_10×10_2 | **588** | **588** | **588** | **588** | **588** |
| tai_10×10_3 | **598** | **598** | **598** | **598** | **598** |
| tai_15×15_1 | **937** | **937** | **937** | **937** | **937** |
| tai_15×15_2 | **918** | **918** | **918** | **918** | **918** |
| tai_15×15_3 | **871** | **871** | **871** | **871** | **871** |
| tai_20×20_1 | **1 155** | **1 155** | **1 155** | **1 155** | **1 155** |
| tai_20×20_2 | **1 241** | **1 241** | **1 241** | 1 242 | **1 241** |
| tai_20×20_3 | **1 257** | **1 257** | **1 257** | **1 257** | **1 257** |

由于 GCN 方法的图结构特性，使其在某些特定实例上表现更加优秀，能够避免局部最优解的问题。因此，GCN 方法相对于传统的元启发式算法具有更强的全局优化能力，能够在大多数情况下获得最优解。

### 4.3 求解效率分析

为评价提出的 GCN 方法在大规模 OSSP 实例中的求解效率，针对 10 种不同规模的 OSSP 随机实例，分别采用 GA、ACO、PSO 和 GCN 进行求解。对每个实例重复进行 10 次测试，在每次测试中都记录从开始读取输入数据直至输出调度结果所耗运行时间（秒），并取 10 次测试的平均运行时间作为求解该实例的计算时间（$t$）。

表 3 展示了采用 GA、ACO、PSO 和 GCN 分别求解 10 种不同规模的 OSSP 随机实例的最佳 Makespan 和计算时间。

表 3 OSSP 随机实例的求解结果

Table 3 Solution results for OSSP random instances

| Instance | GA | | ACO | | PSO | | GCN | |
|---|---|---|---|---|---|---|---|---|
| | *BS* | *t* | *BS* | *t* | *BS* | *t* | *BS* | *t* |
| Test_30_20 | 1 916 | 2 277 | 1 925 | 1 798 | 2 072 | 2 380 | **1 691** | 26.8 |
| Test_40_20 | 2 277 | 2 807 | 2 286 | 2 369 | 2 414 | 3 124 | **2 037** | 35.4 |
| Test_50_20 | 2 871 | 3 295 | 2 879 | 2 950 | 3 123 | 3 506 | **2 652** | 41.2 |
| Test_60_20 | 3 013 | 3 946 | 3 177 | 3 241 | 3 231 | 4 436 | **2 807** | 60.9 |
| Test_70_20 | 3 589 | 4 295 | 3 666 | 3 900 | 3 779 | 4 566 | **3 321** | 72.7 |
| Test_80_20 | 4 072 | 5 269 | 4 367 | 4 267 | 4 511 | 6 410 | **3 756** | 79.3 |
| Test_90_20 | 5 266 | 6 236 | 5 283 | 5 093 | 5 532 | 7 073 | **4 894** | 98.6 |
| Test_100_20 | 5 653 | 6 843 | 5 686 | 5 386 | 6 051 | 7 388 | **5 011** | 116.4 |
| Test_150_20 | 7 033 | 14 458 | 7 582 | 11 686 | 8 124 | 17 885 | **5 833** | 192.0 |
| Test_200_20 | 9 245 | 25 631 | 9 786 | 20 641 | 10 014 | 30 875 | **7 323** | 240.1 |

如表 3 所示，与 GA、ACO、PSO 这 3 种元启发式算法相比，采用 GCN 方法能够在更短的计算时间内得到更佳的 Makespan。这主要得益于 GCN 方法能利用图卷积网络的特性更好地学习 OSSP 的结构信息，以减少搜索空间的大小，从而大大减少了计算时间。此外，GCN 方法通过链路预测能更好地处理 OSSP 中存在的约束条件，从而更好地指导解的生成过程。

## 5 结语

本文提出了一种基于 GCN 求解大规模 OSSP 的方法。首先，使用析取图表示 OSSP 的状态，这能充分利用节点特征信息来求解 OSSP。然后，设计基于 GCN 的开放车间调度模型，这可以帮助模型更好地处理工序节点之间的关系。最后，通过基于图卷积网络的链路预测求解 OSSP，这不仅有利于提高求解大规模 OSSP 的质量，而且还能显著提高求解大规模 OSSP 的效率。针对 OSSP 基准实例和随机生成的 OSSP 实例开展了大量实验，以验证所提出的基于 GCN 求解大规模 OSSP 方法的有效性。实验结果表明，与 3 种元启发式算法相比，所提出的 GCN 方法在 30 个 OSSP 基准实例中具有相同的求解质量，在 10 个大规模 OSSP 随机实例中表现出了更好的求解质量和求解效率。

为了进一步提高大规模 OSSP 的求解效率，下一步将采用 Spark-GPU 平台，对基于 GCN 的开放车间调度方法进行高效分布式并行化，以便更快地找到较优的调度方案。

**参考文献：**

[1] AHMADIAN M M，KHATAMI M，SALEHIPOUR A，et al. Four Decades of Research on the Open-Shop Scheduling Problem to Minimize the Makespan[J]. European Journal of Operational Research，2021，295(2)：399−426.

[2] 陈 祥 . 开放车间调度问题研究及其应用 [D]. 武汉：湖北工业大学，2018.
CHEN Xiang. Research and Application of Open Shop Scheduling Problem[D]. Wuhan：Hubei University of Technology，2018.

[3] WANG W J，TIAN G D，ZHANG H H，et al. A Hybrid Genetic Algorithm with Multiple Decoding Methods for Energy-Aware Remanufacturing System Scheduling Problem[J]. Robotics and Computer-Integrated Manufacturing，2023，81：102509.

[4] 穆亭江 . 开放车间调度问题下融合强化学习的遗传算法研究 [D]. 北京：北京交通大学，2022.
MU Tingjiang. Research on Genetic Algorithm Integrating Reinforcement Learning for Open Shop Scheduling Problem[D]. Beijing：Beijing Jiaotong University，

2022.

[5] ABREU L R，TAVARES-NETO R F，NAGANO M S. A New Efficient Biased Random Key Genetic Algorithm for Open Shop Scheduling with Routing by Capacitated Single Vehicle and Makespan Minimization[J]. Engineering Applications of Artificial Intelligence，2021，104：104373.

[6] RAHMANI H A A，VAHIDI J，SAEMI B，et al. Extended Genetic Algorithm for Solving Open-Shop Scheduling Problem[J]. Soft Computing，2019，23(13)：5099−5116.

[7] 赵小惠，卫艳芳，王凯峰，等 . 改进蚁群算法的柔性作业车间调度问题研究 [J]. 组合机床与自动化加工技术，2022(2)：165−168.
ZHAO Xiaohui，WEI Yanfang，WANG Kaifeng，et al. Research on Flexible Job Shop Scheduling Problem Based on Improved Ant Colony Algorithm[J]. Modular Machine Tool & Automatic Manufacturing Technique，2022(2)：165−168.

[8] KURDI M. Ant Colony Optimization with a New Exploratory Heuristic Information Approach for Open Shop Scheduling Problem[J]. Knowledge-Based Systems，2022，242：108323.

[9] BLUM C. Beam-ACO：Hybridizing Ant Colony Optimization with Beam Search：An Application to Open Shop Scheduling[J]. Computers and Operations Research，2005，32(6)：1565−1591.

[10] 张宇嘉，宋　威 . 双档案粒子群算法求解柔性作业车间调度问题 [J]. 计算机工程与应用，2023，59(11)：294−301.
ZHANG Yujia，SONG Wei. Double Archive Particle Swarm Optimization Solving Flexible Job-Shop Scheduling Problem[J]. Computer Engineering and Applications，2023，59(11)：294−301.

[11] KACHITVICHYANUKUL V，PONGCHAIRERKS P. A Two-Level Particle Swarm Optimisation Algorithm for Open-Shop Scheduling Problem[J]. International Journal of Computing Science and Mathematics，2016，7(6)：575.

[12] SHA D Y，HSU C Y. A New Particle Swarm Optimization for the Open Shop Scheduling Problem[J]. Computers and Operations Research，2008，35(10)：3243−3261.

[13] MARROUCHE W，HARMANANI H M. Heuristic Approaches for the Open-Shop Scheduling Problem[C]// LATIFI S. Information Technology-New Generations. Cham：Springer，2018：691−699.

[14] ABREU L R，PRATA B A，FRAMINAN J M，et al. New Efficient Heuristics for Scheduling Open Shops with Makespan Minimization[J]. Computers & Operations Research，2022，142：105744.

[15] TAILLARD E. Benchmarks for Basic Scheduling Problems[J]. European Journal of Operational Research，1993，64(2)：278−285.

（责任编辑：申　剑）