第 12 章 车间调度问题

车间调度问题（Job Shop Scheduling Problem, JSP）是运筹学与生产管理领域中的经典优化问题，旨在通过合理安排生产资源（如机器、工人、时间等），在满足约束条件的前提下，实现生产效率最大化或成本最小化。其核心挑战在于如何在复杂的工序约束和资源限制下，找到最优或近似最优的调度方案。作为制造系统管理的核心问题之一，其研究不仅具有理论深度，更对实际工业生产具有重要指导意义。

12.1 车间调度问题介绍

车间调度问题的研究起源于 20 世纪初的工业革命时期。随着大规模生产模式的兴起，工厂需要协调多台机器、多个工序和不同订单之间的复杂关系，传统的人工排程方法逐渐难以满足需求。20 世纪 50 年代，随着计算机技术的出现，学者开始尝试用数学建模和算法解决调度问题。1954 年，Johnson 提出的两机器流水车间调度算法成为该领域的里程碑。此后，车间调度问题逐渐发展为运筹学中一个独立的研究方向，并在制造业、物流、服务业等领域得到广泛应用。车间调度问题根据生产环境的不同可分为多种类型：

（1）并行机调度问题（Parallel Machine Scheduling Problem, PMSP）：是经典调度问题的重要分支，其核心目标是为多台可并行工作的机器分配若干待加工作业，在满足机器能力约束的前提下优化生产效率。该问题的典型场景包括半导体晶圆制造、云计算任务分配和纺织生产线等，其中所有机器功能相同（同速机）或处理速度不同（异速机），每个作业只需在一台机器上完成加工且不中断。

（2）作业车间调度（Job Shop Scheduling, JSP）：每个作业包含多个工序，且不同作业的工序顺序可能不同。例如，汽车制造中，不同型号的零件可能需要不同的加工路线。

（3）流水车间调度（Flow Shop Scheduling, FSP）：所有作业的工序顺序一致，但不同机器上的处理时间可能不同，常见于标准化生产线。

（4）开放车间调度（Open Shop Scheduling, OSP）：工序间没有固定顺序约

束，调度自由度更高，但优化难度更大。

实际应用中，车间调度还需考虑动态因素（如机器故障、紧急订单）、多目标优化（如最小化完工时间、平衡机器负载）以及资源约束（如能源消耗、工人技能）等问题。

本章将对其中最为常见的并行机器车间调度问题、柔性作业车间调度问题以及流水车间调度问题分别进行介绍。

12.2 并行机器调度问题

并行机器车间调度问题是调度领域一个经典问题，研究如何将多个作业分配到多台并行机器上加工，以优化生产效率。其特点包括：机器可为同构（处理速度相同）或异构（速度不同）；作业可在任意机器上执行，但处理时间可能不同；优化目标常为最小化总完工时间、最大延迟或平衡机器负载。该问题广泛存在于制造业、云计算任务分配等领域，求解方法涵盖贪心规则、动态规划及元启发式算法（如遗传算法），是资源优化与调度理论的重要研究方向。

12.2.1 并行机器调度问题的数学模型

本节将建立一个以最小化总完工时间为目标的并行机器调度问题的数学模型。最小化完工时间的并行机器调度问题可描述如下：有n个相互独立的作业，使用集合$J = \{1, 2, \ldots, n\}$来表示，有m个机器，且它他们的索引集合被定义为$M = \{1, 2, \ldots, m\}$。作业i在机器j上的处理时间为p_{ij}（若机器同构，则简化为p_{i}）。目标是要找到一个最小调度，即确定每台机器上加工的作业索引，使加工完所有作业的时间最短。接下来定义决策变量：$x_{ij} \in \{0,1\}$：若作业i分配到机器j，则$x_{ij} = 1$，否则为 0；$C_i \geq 0$：作业i的完成时间；$s_i \geq 0$：作业i的开始时间。

问题的优化目标被定义为最小化所有作业的完成时间之和，即

$$\mathrm{Min} \sum_{i=1}^{n} C_i$$

数学模型的约束条件分别为：

（1）作业分配约束，即每个作业必须分配到一台机器：

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i \in J$$

（2）作业完成时间等于开始时间加上处理时间：

$$C_i = s_i + \sum_{j=1}^{m} x_{ij} \cdot p_{ij}, \quad \forall i \in J$$

（3）同一台机器上的作业不能重叠。对任意两台作业i和k，若均分配到机器j，则需满足：

$$s_i + p_{ij} \le s_k \quad 或 \quad s_k + p_{kj} \le s_i$$

为将上述约束转化为线性形式，引入辅助变量$y_{ikj} \in \{0,1\}$：若作业i在机器j上先于作业k加工，则$y_{ikj} = 1$，否则为 0。基于此，添加以下约束（M为极大常数）：

$$s_i + p_{ij} \le s_k + M(1 - y_{ikj}) + M(2 - x_{ij} - x_{kj}), \quad \forall i < k, j$$

$$s_k + p_{kj} \le s_i + M y_{ikj} + M(2 - x_{ij} - x_{kj}), \quad \forall i < k, j$$

该模型为混合整数线性规划（MILP）问题，适用于理论分析与启发式算法设计。

例题 12.1 某工厂有 3 台并行机器(M_1, M_2, M_3)，需要加工 5 个独立工件$(J_1, J_2, J_3, J_4, J_5)$。每个工件只需在一台机器上加工一次，且加工时间不可中断。各工件的加工时间分别为 4、5、2、6、3。目标是合理分配工件到机器上，使得所有工件的总完成时间（即最后一台机器完成其分配任务的时刻，称为最大完工时间）最小化。请给出一种可行解，并计算其最大完工时间。

解 步骤 1：分配方案设计

使用启发式方法（如最长加工时间优先，LPT）分配工件：

（1）按加工时间降序排列工件：$(J_4, J_2, J_1, J_5, J_3)$。

（2）依次将每个工件分配到当前负载最小的机器：

- 初始负载：$(M_1 = 0, M_2 = 0, M_3 = 0)$。

- 分配J_4：选择M_1，负载变为$M_1 = 6$。

- 分配J_2：选择M_2，负载$M_2 = 5$。

- 分配J_1：选择M_3，负载$M_3 = 4$。

- 分配J_5：选择当前负载最小的M_3，负载$M_3 = 4 + 3 = 7$。

- 分配J_3：选择负载最小的M_2，负载$M_2 = 5 + 2 = 7$。

最终分配：

- M_1: J_4 → 完成时间 6

- M_2: J_2, J_3 → 完成时间 5+2=7

- M_3: J_1, J_5→ 完成时间 4+3=7

步骤 2：计算最大完工时间

$$C_{\max} = \max\{6,7,7\} = 7$$

12.2.2 并行机器调度问题的求解方法

当作业数量较多时，一个可行的解决方案是使用元启发式算法来求出一个近似最优解。其中，遗传算法是最为常用的算法之一。在并行机器调度问题中，通过合理的编码、适应度函数和遗传操作设计，可以有效找到近似最优解。以下是针对并行机器调度问题的遗传算法设计思路：

（1）编码设计

编码方式采取基于机器作业列表的编码。染色体表示为每台机器上的作业顺序列表，确保每个作业仅分配一次。例如，对于 3 台机器（M1, M2, M3）和 5 个作业（J1-J5），染色体结构可表示为[[J1, J3], [J2, J5],[J4]]，其中内层的三部分分别表示机器 M1 的工作顺序，机器 M2 的工作顺序和机器 M3 的工作顺序。

初始化种群。分配作业到机器：每个作业随机分配到一台机器；生成作业顺

序：每台机器上的作业随机排列。例如，父代 1 为[[J1, J3], [J2, J5], [J4]]，父代 2 为[[J5], [J1, J4], [J2, J3]]。

（2）交叉操作与修复机制

交叉操作采用基于机器的交叉：交换父代的某台机器作业列表，可能导致作业重复或遗漏。步骤如下：①随机选择一台机器（如 M1）；②子代 1 继承父代 1 的 M1 作业列表，父代 2 的其余作业分配；③子代 2 继承父代 2 的 M1 作业列表，父代 1 的其余作业分配。例如父代 1 的 M1 作业为[J1, J3]，父代 2 的 M1 作业为[J5]。交叉后的子代 1 为[[J1, J3], [J4], [J2, J5]]（J5 重复出现在 M1 和 M3），子代 2 为[[J5], [J2, J5], [J1, J3, J4]]（J5 重复出现在 M1 和 M2）。

因为交叉过程中可能发生作业重复和遗漏的现象，导致编码不可行，因此通过下面的步骤修复不可行解。①检测重复作业：遍历所有机器，记录每个作业出现的次数。②移除重复作业：对于重复作业，随机保留一台机器中的实例，从其他机器中删除。③补全遗漏作业：检查未分配的作业，随机分配到某台机器。一个修复的示例为子代 1 修复前：[[J1, J3], [J4], [J2, J5]]（J5 重复）。修复后为随机保留 M3 的 J5，移除 M1 的 J5，得到[[J1, J3], [J4], [J2, J5]]，并无重复。

（3）变异操作与可行性保持

变异类型的主要分为以下两种：①机器变异：随机选择一个作业，移动到另一台机器。原机器中移除该作业，目标机器中随机插入；②顺序变异：随机选择一台机器，交换其作业顺序。例如原始染色体为[[J1, J3], [J2, J5], [J4]]，机器变异（移动 J3 到 M2）为[[J1], [J2, J5, J3], [J4]]，顺序变异（交换 M2 的 J5 和 J3）为[[J1], [J2, J3, J5], [J4]]

（4）与其他问题的区别

①编码复杂性。经典装箱问题仅需机器分配，而并行机器调度问题若存在顺序依赖，需额外编码作业顺序。如旅行商问题仅需排列，并行机器调度问题需多台机器的排列组合。

②修复机制需求。交叉操作易导致作业重复/遗漏，需显式修复。如流水车间调度仅需排列，无重复作业问题。

（5）总结

编码：染色体表示每台机器的作业顺序列表，确保全局唯一性。交叉修复：检测并移除重复作业，补全遗漏作业。变异：原子操作保证可行性，无需额外修复。通过上述设计，遗传算法可有效探索解空间，同时处理机器分配和作业顺序的耦合关系，适用于具有复杂约束的并行机器调度问题

12.3 柔性作业车间调度问题

柔性作业车间调度问题（Flexible Job-Shop Scheduling Problem, FJSP）是传统作业车间调度问题的扩展。在柔性作业车间调度问题中，每个工件的加工工序不再限定于单一机器，而是可以在多个可选机器中选择，且不同机器对同一工序的处理时间可能不同。优化目标通常包括最小化最大完工时间、延迟或设备负载均衡等。其核心挑战在于既要分配工序到机器，又要确定各机器上的加工顺序。柔性作业车间是作业车间与并行机环境的结合体，每个作业具有一道或多道工序（作业之间的工序可能不同），且每一道工序可在一台或多台并行机器上加工。在航空航天制造中，零件（如涡轮叶片）需经过多道精密工序（如锻造、热处理、数控加工、质检），且每道工序可能有多种可选设备：数控加工阶段可使用多台不同型号的 CNC 机床，根据刀具配置和当前负载动态选择；热处理可在不同炉温控制的设备中进行。通过 FJSP 模型，工厂能灵活分配工序至合适设备，减少瓶颈工序等待时间，同时平衡设备负载，最终缩短交付周期并降低成本。

12.3.1 柔性作业车间调度问题的数学模型

本小节介绍柔性作业车间调度问题的典型数学模型，以最小化最大完工时间为目标，采用混合整数线性规划形式描述。首先定义工件集合为$J = \{1, 2, \dots, n\}$。其中，工件j的工序集合为$O_j = \{o_{j1}, o_{j2}, \dots, o_{jm_j}\}$；其次，定义机器集合为$M = \{1, 2, \dots, k\}$，并且工序$o_{ji}$的可选的机器集合为$M_{ji} \subseteq M$。$p_{jik}$表示工序$o_{ji}$在机器$k$上的加工时间（若$k \notin M_{ji}$，则$p_{jik} = +\infty$）。接下来定义决策变量：$x_{jik} \in \{0,1\}$：若工序$o_{ji}$分配到机器$k$，则$x_{jik} = 1$，否则为 0；$s_{ji} \geq 0$：工序$o_{ji}$的开始时间；$C_{\max} \geq 0$：最大完工时间。$y_{ji,j'i'k} \in \{0,1\}$：若在机器$k$上，工序$o_{ji}$在$o_{j'i'}$之后加工，则

$y_{ji,j'i'k} = 1$，否则为 0。

问题的优化目标被定义为最小化所有作业的完成时间之和，即

$$\min C_{\max}$$

数学模型的约束条件分别为：

（1）工序分配约束：每个工序必须分配到一台可选机器，即

$$\sum_{k \in M_{ji}} x_{jik} = 1, \forall j \in J, \forall o_{ji} \in O_j$$

（2）工序顺序约束：同一工件的相邻工序需顺序加工（前序工序完成后，后续工序才能开始），即

$$s_{j(i+1)} \geq s_{ji} + \sum_{k \in M_{ji}} x_{jik} \cdot p_{jik}, \forall j \in J, \forall i < m_j$$

（3）机器加工冲突约束：同一机器上的任意两个工序不可重叠加工。对于分配到同一机器的两工序o_{ji}和$o_{j'i'}$，$\forall j, j' \in J, o_{ji} \in O_j, o_{j'i'} \in O_{j'}$，需满足

$s_{ji} \geq s_{j'i'} + \sum_{k \in M_{j'i'}} x_{j'i'k} \cdot p_{j'i'k} - \Gamma \cdot \left(1 - y_{ji,j'i'k}\right)$;

$s_{j'i'} \geq s_{ji} + \sum_{k \in M_{ji}} x_{jik} \cdot p_{jik} - \Gamma \cdot y_{ji,j'i'k}$;

$y_{ji,j'i'k} \in \{0,1\}$;

其中Γ为足够大的正数（大 M 法）。

（4）完工时间约束:总完工时间需覆盖所有工序的完成时间：

$$C_{\max} \geq s_{ji} + \sum_{k \in M_{ji}} x_{jik} \cdot p_{jik}, \forall j \in J, \forall o_{ji} \in O_j$$

以上模型具有以下特点。首先，其约束数量随工序和机器数呈指数增长，尤其析取约束导致模型求解困难。其次，上述模型适合小规模问题，大规模实例需结合启发式算法（如遗传算法、禁忌搜索）。最后，上述模型可额外地引入设备

负载均衡、交货期惩罚等目标或约束来满足现实需要。

例题 12.2 某工厂需加工 2 个工件(J_1, J_2)，每个工件包含 2 道工序，每道工序可在多台机器上加工，且不同机器的加工时间不同。具体信息如下：

令O_{11}表示工件J_1的第一个工序，其可选机器 M_1（加工时间 3 小时）或M_2（加工时间 4 小时），使用W_{11}表示工序O_{11}的可选机器集合。使用O_{12}表示工件J_1的第二个工序，其可选机器M_2（加工时间 2 小时）或M_3（加工时间 5 小时），使用W_{12}表示工序O_{12}的可选机器集合。工序顺序：$O_{11} \to O_{12}$。

使用O_{21}表示工件J_2的第一个工序，其可选机器M_1（加工时间 2 小时）或M_3（加工时间 3 小时），使用W_{21}表示工序O_{21}的可选机器集合。使用O_{22}表示工件J_2的第二个工序，其可选机器M_2（加工时间 4 小时）或M_3（加工时间 1 小时），使用W_{22}表示工序O_{22}的可选机器集合。工序顺序：$O_{21} \to O_{22}$。

目标是为每道工序选择机器并安排加工顺序，使得所有工件的最大完工时间最小。请建立一个数学规划模型来实现上述目标。

解 首先，定义以下决策变量。$x_{ijk} \in \{0,1\}$：若工序O_{ij}分配到机器M_k，则$x_{ijk} = 1$，否则为 0。$s_{ij} \geq 0$：工序O_{ij}的开始时间。$c_{ij} \geq 0$：工序O_{ij}的完成时间。$C_{\max} \geq 0$：最大完工时间。

根据题目的要求，将目标函数设置为

$$\text{Min} C_{\max}$$

最后，设立以下约束条件以保证获得的解决方案的可行性

（1）工序必须分配到一台可用机器：

$$\sum_{k \in W_{ij}} x_{ijk} = 1, \forall i, j$$

（2）同一工件的工序顺序约束：

$$s_{i(j+1)} \geq c_{ij} \quad \forall i, j$$

（3）机器独占性约束（同一机器的工序不能重叠）：

$$若x_{ijk} = x_{lmn} = 1 且k = n,则s_{ij} \geq c_{lm} 或 s_{lm} \geq c_{ij}.$$

（4）完成时间定义

$$c_{ij} = s_{ij} + \sum_k p_{ijk}x_{ijk} \forall i,j,$$

其中p_{ijk}为工序O_{ij}在机器M_k的加工时间。

（5）总完成时间约束：

$$C_{max} \geq c_{ij} \forall i,j,$$

12.3.2 求解柔性作业车间调度问题的遗传算法

在现实生产环境中，柔性作业车间调度问题通常采用启发式算法而非精确算法，主要原因如下：

（1）计算时间可接受

精确算法（如分支定界）的求解时间随问题规模呈指数级增长。例如，一个包含 50 道工序、10 台机器的 FJSP 可能需要数小时甚至数天才能找到最优解。实际车间调度需在几分钟到几小时内响应动态变化（如紧急插单、机器故障）。启发式算法（如遗传算法、禁忌搜索）能在数秒至数分钟内给出可行解，避免因等待优化结果导致生产线停滞。

（2）处理大规模问题的能力

柔性作业车间调度问题是典型的 NP-hard 问题，当工序数超过 100 或机器数超过 20 时，精确算法几乎无法求解。实际车间往往涉及数百道工序（如汽车装配线、电子产品制造），启发式算法通过局部搜索或群体智能（如粒子群优化）高效探索解空间，即使无法保证最优，也能提供高质量调度方案。

（3）适应动态变化与不确定性

车间常面临机器故障、订单变更、物料延迟等突发情况。启发式算法能快速调整现有解（如重新分配故障机器的工序），而精确算法需从头重新计算，难以

满足实时性。启发式算法可融入鲁棒性目标（如缓冲时间分配），生成对扰动不敏感的调度方案，而精确算法对此类复杂目标建模困难。

（4）多目标优化灵活性

生产调度需同时优化完工时间、设备负载均衡、能耗、交货期等多目标。精确算法需通过加权或分层处理，而启发式算法（如 NSGA-II）可直接生成 Pareto 前沿解集，支持决策者权衡选择。半导体工厂需在最小化生产周期的同时，避免某些精密设备过载。基于遗传算法的多目标优化能快速提供多种折衷方案。

（5）资源与成本限制

精确算法依赖高性能计算资源（如商用求解器 CPLEX），而启发式算法可在普通服务器甚至边缘设备上运行，适合中小企业。启发式算法的模型更易调整（如添加新约束时无需重构数学规划），且开源框架（如 DEAP、Optapy）降低了开发门槛。为此，本书提供以下关于柔性作业车间调度问题的遗传算法设计思路及 Python 实现代码（请见电子资源）。算法以最小化最大完工时间为目标。

（1）编码设计：双层编码机制

柔性作业车间调度问题的染色体需同时编码机器分配和工序顺序，常用双层编码（两段式编码）。第一段（机器分配段）：表示每道工序选择的机器编号；第二段（工序顺序段）：表示所有工序的全局加工顺序，需满足同一作业的工序顺序约束。

具体实现的示例：3 个作业（Job1-3），每个作业有 2 道工序（O11, O12; O21, O22; O31, O32），机器集合 M1-M4。

机器分配段：直接编码每道工序的机器选择。例如[M3, M1, M2, M4, M1, M3]，其分别对应工序 O11, O12, O21, O22, O31, O32 被分配的工序。

工序顺序段：使用基于工序优先级的排列编码，需隐式满足顺序约束。例如[O11, O21, O31, O12, O22, O32]，其中作业 1 的工序 O12 必须在 O11 之后。

编码初始化。机器分配初始化：对每道工序，从其候选机器集合中随机选择一台。工序顺序初始化：生成全局工序序列，确保同一作业的工序顺序不颠倒（如

O11 始终在 O12 之前）。

（2）交叉操作与不可行解修复

交叉操作需同时作用于机器分配段和工序顺序段，可能破坏工序顺序约束或机器选择约束，需针对性修复。

交叉操作设计。机器分配段的交叉：采用单点交叉或均匀交叉，直接交换父代染色体片段。风险：可能选择不属于候选机器集合的机器（如 M5）。修复：对非法机器编号，从该工序的候选机器集合中随机替换。工序顺序段的交叉：采用保持优先顺序交叉(POX)，具体步骤如下：

步骤 1：选择一个作业的子集（如 Job1 和 Job3）；

步骤 2：子代 1 继承父代 1 中属于这些作业的工序顺序，其余工序按父代 2 的顺序填充；

步骤 3：子代 2 反向操作。

上述保持优先顺序交叉可以保证同一作业的工序顺序不颠倒，避免显式修复。例如：父代 1 为[O11, O21, O31, O12, O22, O32]，父代 2 为[O21, O11, O31, O22, O12, O32] 。选择 Job1 和 Job3：

子代 1 继承父代 1 中的 O11, O12, O31, O32，剩余位置按父代 2 顺序填充 O21, O22 → [O11, O21, O31, O12, O22, O32]

子代 2 继承父代 2 中的 O21, O22, O31, O32，剩余位置按父代 1 顺序填充 O11, O12 → [O21, O11, O31, O22, O12, O32]

表 12.1 遗传算法求解柔性作业车间问题不可行解的修复

不可行类型	修复方法
工序顺序冲突（如 O12 在 O11 前）	使用拓扑排序调整顺序，确保同一作业的工序顺序约束
机器选择非法（如 M5）	从该工序的候选机器集合中随机选择合法机器。

机器负载不均衡	动态调整机器分配段，将高负载机器的工序迁移到低负载机器（需满足候选集合约束）。

（3）变异操作与可行性保持

变异需在局部扰动解的同时保证解的可行性。常见的变异类型分为机器分配变异和工序顺序变异。前者指随机选择一道工序，从其候选机器集合中重新选择一台机器（即使加工时间更长）。例如将 O11 的机器从 M3 变为 M2（需 M2 在候选集合中）。后者则包括，相邻交换指交换同一机器上相邻且无顺序约束的工序。插入变异指将某工序插入到同一机器加工序列的合法位置（不违反作业内顺序）。

变异示例：

原始工序顺序：[O11, O21, O31, O12, O22, O32]

插入变异：将 O12 插入到 O21 之后 → [O11, O21, O12, O31, O22, O32]（需检查 Job1 的 O12 是否在 O11 之后）。

（4）算法流程总结

①种群初始化：生成满足机器候选约束和工序顺序约束的初始解。

②适应度评估：通过仿真计算每个染色体的最大完工时间。

③选择操作：采用锦标赛选择或轮盘赌选择保留优质个体。

④交叉与修复：使用保持优先顺序交叉和机器分配修复机制生成子代。

⑤变异与验证：对子代进行局部变异并检查可行性。

⑥迭代优化：直到达到最大迭代次数或收敛。

通过上述设计，遗传算法可有效处理柔性作业车间调度问题中机器选择与工序顺序的复杂耦合，在可行解空间中高效搜索全局最优解。

12.4 流水车间调度问题

不同于并行机器调度问题中多个相同机器处理独立、单工序的任务（如多个收银台处理顾客结账），也不同于柔性作业车间调度问题工件有各自工序顺序，

且每道工序可选多台机器加工（如定制零件在不同车床、铣床间灵活选择加工路线），流水车间是生产调度中一类典型的环境，其特点是所有工件按照相同的工艺路线依次经过多台机器。例如，在汽车装配线中，车身可能需要依次经过焊接、喷漆、组装等多个固定工序。流水车间调度问题的核心在于确定工件的加工顺序，以最小化最大完工时间、延迟时间或其他性能指标。

12.4.1 流水车间调度问题介绍

流水车间调度问题（Flow Shop Scheduling Problem, FSSP）可描述为：给定n个工件和m台机器，每个工件需依次在机器$(1,2,\dots,m)$上加工。每个工件在每台机器上的加工时间已知。每台机器同一时间只能处理一个工件，且工件一旦开始加工不能中断。目标是找到工件的加工顺序，使得最后一个工件在所有机器上的完成时间最短（即最小化最大完成时间$C_{\max}$）。流水车间调度问题属于 NP 难问题，当机器数量$m \geq 3$时，不存在多项式时间的精确算法。因此，目前主要依赖启发式算法或元启发式算法（如遗传算法、模拟退火）进行求解。流水车间调度广泛应用于半导体制造、化工生产、食品加工等领域。例如，在芯片制造中，晶圆需依次经过光刻、蚀刻、清洗等工序，优化调度可显著降低生产成本。

12.4.2 流水车间调度问题的数学模型

下面建立一个关于流水车间调度问题的数学模型，目的在于获得能够使最大完成时间最小化的生产计划。首先，定义工件集合为$J = \{1,2,\dots,n\}$，机器集合为$M = \{1,2,\dots,m\}$。使用p_{ij}表示工件i在机器j上的加工时间，并使用C_{ij}:表示工件i在机器j上的完成时间。需要考虑的约束条件如下所示：

（1）顺序约束：每个工件必须按机器顺序$1 \to 2 \to \cdots \to m$加工。

（2）机器独占性：同一时刻一台机器只能加工一个工件。

（3）工件连续性：工件i在机器$j+1$的加工必须在其在机器j的加工完成后开始。

模型的目标是最小化最大完成时间，即

$$\text{Min}C_{\max} = \max_{i \in J} C_{im}$$

约束条件分别表达如下：

（1）机器加工顺序：

$$C_{i1} \geq p_{i1} \quad \forall i \in J$$

（2）工件连续性：

$$C_{i,j} \geq C_{i,j-1} + p_{ij} \quad \forall i \in J, j \geq 2$$

（3）机器独占性：

$$C_{i,j} \geq C_{k,j} + p_{ij} \text{或} C_{k,j} \geq C_{i,j} + p_{kj}$$

（若工件i和k在同一机器j上加工，且i在k之后加工）

例题 12.3 某工厂采用流水车间生产模式，需加工 3 个工件（J_1, J_2, J_3），每个工件需依次经过 3 台机器（M_1, M_2, M_3）加工，各工件在不同机器上的加工时间如表 12.2 所示。

表 12.2 例题 12.3 中的加工时间

机器 / 工件	M_1	M_2	M_3
J_1	2	3	2
J_2	4	1	3
J_3	1	2	4

请确定工件的加工顺序，使得所有工件的最大完工时间最小，并且通过枚举法验证所有可能顺序，找到最优解。

解 步骤 1：枚举所有可能的加工顺序

3 个工件共有$3! = 6$种排列方式，具体排序如下所示：

（1）顺序$J_1 \to J_2 \to J_3$；

（2）顺序$J_1 \to J_3 \to J_2$；

（3）顺序$J_2 \to J_1 \to J_3$；

（4）顺序$J_2 \to J_3 \to J_1$；

（5）顺序$J_3 \to J_1 \to J_2$；

（6）顺序$J_3 \to J_2 \to J_1$。

步骤 2：分别计算以上 6 种顺序的最大完工时间：

以顺序$J_3 \to J_1 \to J_2$为例，计算各机器所有工件的顺序完成时间：

对于机器M_1，工件J_3的完成时间为1，工件J_1的完成时间为3，工件J_2的完成时间为7。

对于机器M_2，工件J_3的完成时间为3（需等待J_3在M_1完成），工件J_1的完成时间为6（需等待J_1在M_1完成），工件J_2的完成时间为 8（需等待J_2在M_1完成）。

对于机器M_3，工件J_3的完成时间为7（需等待J_3在M_2完成），工件J_1的完成时间为9（需等待J_1在M_2完成），工件J_2的完成时间为 12（需等待J_2在M_2完成）。

综上所示，总完成时间为 12，由J_2在M_3的完成时间决定。

同理，我们可以计算其他顺序的最大完工时间，具体结果如下所示：

（1）顺序$J_1 \to J_2 \to J_3$:14;

（2）顺序$J_1 \to J_3 \to J_2$:13;

（3）顺序$J_2 \to J_1 \to J_3$:15;

（4）顺序$J_2 \to J_3 \to J_1$:16;

（5）顺序$J_3 \to J_1 \to J_2$:12;

（6）顺序$J_3 \to J_2 \to J_1$:14.

答案总结:最优加工顺序为$J_3 \to J_1 \to J_2$，对应的最小最大完工时间为 12。

12.4.3 遗传算法求解流水车间调度问题

本节设计一个遗传算法来求解 10.4.2 中建模的流水车间调度问题，具体包括

以下步骤：

（1）编码设计

流水车间调度问题的解本质是确定所有作业的全局加工顺序，且该顺序在所有机器上保持一致。因此，染色体编码为作业的排列序列。染色体结构：一个长度为 N（作业总数）的排列，表示作业的加工顺序。例如使用[3, 1, 4, 2]来表示作业顺序为 Job3 → Job1 → Job4 → Job2。此外，还需要满足合法性要求：排列中每个作业编号唯一且完整，无重复或缺失。编码初始化。随机排列生成：对作业编号进行随机排列，确保每个作业出现且仅出现一次。

（2）交叉操作与不可行解修复

由于流水车间调度问题的编码为排列，交叉操作需生成合法的排列子代。常用顺序交叉（OX）或部分匹配交叉（PMX），这些方法天然保证子代的合法性，无需额外修复。

顺序交叉的步骤包括：

①随机选择父代 1 的一个连续子序列（如父代 1 的索引 1-2：[1, 4]）。

②子代 1 保留父代 1 的子序列，剩余位置按父代 2 的顺序填充，跳过子序列中已有的作业。

③子代 2 反向操作。示例：父代 1 为[3, 1, 4, 2]，父代 2 为[2, 4, 1, 3]。

选择父代 1 的子序列[1, 4]，子代 1 保留[1, 4]，剩余位置按父代 2 顺序填充（跳过 1 和 4）→[2, 1, 4, 3]。子代 2 保留父代 2 的子序列（假设选[4, 1]），填充父代 1 的剩余作业→[3, 4, 1, 2]。

部分匹配交叉的步骤包括：

①随机选择两个交叉点，交换父代 1 和父代 2 的中间段。

②通过映射关系解决冲突，确保子代为合法排列。

顺序交叉与部分匹配交叉的天然合法性。由于这两种交叉方法直接维护排列的唯一性，生成的子代无需修复。

其他交叉方法（如单点交叉）：若使用不维护排列的交叉方式（如直接交换片段），会导致作业重复或缺失。此时需通过去重补缺修复：

①遍历子代，记录重复或缺失的作业。

②删除重复项，补全缺失项，随机排列剩余位置。

（3）变异操作与可行性保持

变异操作需在局部扰动解的同时保持排列合法性，常用方法如下：

①交换变异（Swap Mutation）：随机选择两个位置交换作业。例如原始：[3, 1, 4, 2] → 交换位置 1 和 3 → [3, 4, 1, 2]。

②插入变异（Insertion Mutation）：随机选择一个作业插入到新位置。例如原始：[3, 1, 4, 2] → 将作业 4 插入到位置 0 → [4, 3, 1, 2]。

③逆转变异（Inversion Mutation）：随机选择一个子序列并反转。例如原始：[3, 1, 4, 2] → 反转位置 1-3 → [3, 2, 4, 1]。

合法性保证：上述操作均不破坏排列的唯一性和完整性，无需修复。

（4）算法流程总结

①初始化种群：生成 N 个随机排列，每个排列表示一个合法的作业顺序。

②适应度计算：通过仿真计算每个排列的最长完成时间。

③选择操作：使用轮盘赌或锦标赛选择保留高质量解。

④交叉操作：应用顺序交叉或部分匹配交叉生成子代，确保合法性。

⑤变异操作：对子代进行交换、插入或逆转变异。

⑥迭代优化：重复选择-交叉-变异直至收敛或达到最大迭代次数。

通过上述设计，遗传算法能够高效搜索流水车间调度问题的最优作业顺序，充分利用排列编码的特性，避免复杂修复逻辑，适用于大规模流水车间调度问题实例。

12.5 本章小结

车间调度作为运筹学在生产管理中的核心应用之一，通过优化资源分配与任务排序，显著提升制造系统的效率与响应能力。本章系统探讨了车间调度的三类经典问题——并行机器车间调度，柔性作业车间调度以及流水车间调度，从问题定义、数学模型构建到求解方法，逐步揭示了其理论与应用价值。车间调度问题的研究不仅推动了运筹学、组合优化和算法设计的发展，还为智能制造、工业 4.0 提供了关键技术支撑。当前的研究趋势聚焦于动态实时调度、多目标协同优化以及人机协同调度，旨在应对个性化定制生产和绿色制造的新需求。

车间调度不仅是理论课题，更是连接运筹学与工业实践的桥梁。随着制造系统向智能化、柔性化演进，调度优化将持续成为提升企业竞争力的关键引擎。建议读者结合开源工具（如 Python 的 SimPy 库）进行仿真实验，并通过案例研究深化对复杂约束与多目标权衡的理解。

研学互通

为进一步深化对车间调度问题的理解，强化理论方法与工程实践的融合性，本模块精选领域内代表性文献。通过研读这些文献，读者可在掌握基础理论框架的同时，把握该问题的学术演进路径与前沿突破，为后续科研探索与技术革新奠定方法论基础。

(1) Pinedo, M. (2016). Scheduling: Theory, algorithms, and systems (5th ed.). Springer.

本书构建了完整的调度理论体系，系统阐释了流水车间、作业车间等经典调度模型的数学建模方法。针对 NP 难特性，深入解析了启发式规则（如 SPT、EDD）、元启发式算法（禁忌搜索、遗传算法）及精确求解技术（分支定界法）的工程实现路径。作为调度领域权威教材，其提出的多目标优化框架与鲁棒性分析方法，为工业级调度系统开发提供了方法论基础，被全球学者和工程师广泛引用。

(2) Garey, M. R., & Johnson, D. S. (1979). Computers and intractability: A guide to the theory of NP-completeness. W. H. Freeman.

该著作通过严格的数学归约，首次证明作业车间调度、流水车间调度等典型问题属于 NP 完全难题。其建立的复杂度分类标准（如 NP-hard、强 NP-hard）揭示了调度问题在多项式时间内精确求解的不可行性，从根本上解释了传统优化方法的局限性。这一理论突破直接推动了后续遗传算法、模拟退火等智能优化算法在调度领域的大规模应用。

(3) Cowling, P., & Johansson, M. (2002). Using real time information for effective dynamic scheduling in manufacturing. International Journal of Production Research, 40(15), 4039–4052.

本文开创性提出基于实时数据驱动的动态调度框架。通过构建事件触发机制（如机器故障、紧急插单）和滚动时域优化模型，实现调度方案在线动态重构，为柔性制造系统应对不确定性扰动提供了可落地的技术范式。

(4) Zhang, H., & Gen, M. (2005). Multistage-based genetic algorithm for flexible job-shop scheduling. In Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics (pp. 2182–2188). IEEE.

针对柔性作业车间调度的复杂耦合约束，设计多阶段协同进化遗传算法。首创工序链编码与机器分配矩阵的双层染色体结构，结合自适应交叉算子和禁忌搜索局部优化，有效解决工序顺序与资源分配的强耦合优化难题。

(5) Rahmani, D., & Ramezanian, R. (2016). A stable reactive approach in dynamic flexible job shop scheduling with unexpected disruptions: A case study. Computers & Industrial Engineering, 98, 360–372.

本研究融合鲁棒优化与反应式调度策略，构建扰动传播控制模型。通过设计稳定性评价函数（方案偏移度 $\Delta<5\%$）和重调度阈值机制，在应对设备故障时仅局部调整受影响工序。

思行经世：我国车间调度技术的应用创新与产业实践

我国作为全球制造业第一大国，车间调度技术已成为推动制造业智能化升级的核心支撑。在"中国制造 2025"战略引导下，该技术通过深度优化生产资源配置、

突破设备效能瓶颈，显著提升了汽车、电子、装备等支柱产业的数字化转型水平，形成了兼具效率与韧性的中国特色应用模式。

在汽车制造领域，广汽、比亚迪等企业采用基于大数据分析的智能调度系统，实现焊装、涂装、总装三大工艺的全流程协同排产。通过动态调整工位节拍与物料配送频次，某车企将混线生产切换时间缩短 40%，在人工成本零增长前提下实现产能提升 15%。电子行业中，华为、富士康等企业运用多目标柔性调度算法，精准应对手机零部件多品种、小批量生产需求，某工厂通过毫秒级实时调度使设备利用率从 68%跃升至 85%，同时降低在制品库存 30%。装备制造业则聚焦重型机械的复杂工艺约束，三一重工引入高精度数字孪生驱动的调度模型，结合应力形变仿真数据，将大型构件加工周期压缩 22%。

我国产学研结合探索出独特技术路径：

（1）工业互联网+调度优化：海尔 COSMOPlat 平台构建端边云架构，集成设备状态感知与调度决策，实现跨车间生产链智能动态重组，订单响应速度提高 25%；

（2）AI 算法突破：清华大学团队开发的融合强化学习的混合量子遗传算法，在某航天部件车间突破组合优化瓶颈，将调度求解速度提升 10 倍；

（3）边缘计算应用：徐工机械在 5G 边缘节点部署分布式轻量化调度引擎，通过计算卸载技术使实时排产响应延迟稳定低于 50ms。

相较于欧美偏重理论最优解，我国车间调度更强调复杂约束下的实用性：既需应对劳动力密集产线的高频动态扰动（如人员轮岗、紧急插单），又需兼容新旧设备混用的自动化岛与人工工位异构协同。当前瓶颈在于设备层、控制层、计划层的多源数据融合度不足，约 60%中小企业的调度仍依赖粗放型经验规则。未来，随着数字孪生技术的规模化普及，车间物理系统与信息空间的闭环深度交互将催生具备自学习能力的新一代自主决策调度体系，在工业互联网平台赋能下进一步释放智能制造潜力。

习题

习题 12.1 流水车间调度问题。某流水车间需要加工 2 个工件(J_1,J_2)，这些工件需要按顺序共经历 3 台机器$(M_1 \to M_2 \to M_3)$，各个工件在各机器的加工时间如下表：

表 12.3 习题 12.3 中的加工时间

机器 / 工件	M_1	M_2	M_3
J_1	2	3	1
J_2	1	2	4

若按顺序$J_1 \to J_2$加工，求最大完工时间。

习题 12.2 作业车间调度问题。现有两个工件，每个工件在两个机器上各有两个工序，具体细节如下所示。

工件J_1的工序为O_{11}（M_1，时间 2）$\to O_{12}$（M_2，时间 3）。

工件J_2的工序为O_{21}（M_2，时间 1）$\to O_{22}$（M_1，时间 4）。

请给出一种可行调度方案，确定每个机器上各工件的加工顺序并计算最大完工时间。

习题 12.3 并行机调度问题。某零件加工厂现在需要加工 3 个工件，他们都有且只有一个相同的工序，加工时间分别为 4、5、2。该工厂有 2 台并行机器可供使用。请使用最长加工时间优先（LPT）规则来分配工件并计算最大完工时间。

习题 12.4 调度规则应用。现在有 4 个工件需在一台机器上加工，他们的加工时间分别为 3, 1, 4, 2。请使用最短加工时间优先（SPT）规则对这些工件进行排序并计算平均流程时间（所有工件完工时间的平均值）。

习题 12.5 数学建模问题。请将涉及 2 台机器以及 3 个工件的并行机调度问题建模为一个整数线性规划问题，写出决策变量、定义目标函数（最小化最大完

工时间）并建立约束。请将建立好的数学模型使用 Python 调用 Gurobi 求解。