中图分类号：　　单位代号：10280

密　　级：　　学　　号：17722061

上海大学

硕士学位论文

SHANGHAI UNIVERSITY

MASTER'S DISSERTATION

题目	一致并行机调度的两阶段鲁棒优化模型与算法研究

作　　者　谢韩鑫

学科专业　系统工程

导　　师　王冰 教授

完成日期　二〇二一年五月

姓　　名：谢韩鑫　　学号：17722061

论文题目：一致并行机调度的两阶段鲁棒优化模型与算法研究

上海大学

本论文经答辩委员会全体委员审查,确认符合上海大学硕士学位论文质量要求。

答辩委员会签名：

主任：

委员：

导　　师：

答辩日期：

姓　　名：谢韩鑫　　学号：17722061

论文题目：一致并行机调度的两阶段鲁棒优化模型与算法研究

原 创 性 声 明

本人声明：所呈交的论文是本人在导师指导下进行的研究工作。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已发表或撰写过的研究成果。参与同一工作的其他同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

密级：

保密期限：

签　名：_____日 期：_____

本论文使用授权说明

本人完全了解上海大学有关保留、使用学位论文的规定，即：学校有权保留论文及送交论文复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容。

（**保密的论文在解密后应遵守此规定**）

签 名：_____导师签名：_____日期：_____

上海大学工学硕士学位论文

一致并行机调度的两阶段鲁棒优化模型和算法研究

姓　　名：谢韩鑫

导　　师：王冰

学科专业：系统工程

上海大学机电工程与自动化学院

二零二一年五月

A Dissertation Submitted to Shanghai University for the Degree of Master in Engineering

Two-stage Robust Optimization Model and Algorithm for Identical Parallel Machine Scheduling

MA Candidate：Xie Hanxin

Supervisor：Wang Bing

Major：System Engineering

School of Mechatronic Engineering and Automation,

Shanghai University

May, 2021

摘　要

生产调度问题是企业生产制造的核心问题之一。通过求解生产调度问题来制定合适的生产计划可以保证生产加工的稳定性和效率。并行机调度问题（Parallel Machine Scheduling Problem, PMSP）是一种经典的生产调度问题，也是联系不同调度问题的纽带之一。而一致并行机调度问题（Identical Parallel Machine Scheduling, IPMS）又是最普遍的一种并行机调度类型，广泛地存在于各种智能制造领域。因而对一致并行机调度问题的研究有着重要的学术意义和实际价值。

目前对于一致并行机调度问题的研究主要集中在确定性问题上，但实际的生产环境中充斥着各种不确定因素。这使得通过求解传统确定性模型得到的生产方案难以达到生产预期。因此研究不确定环境下的一致并行机调度问题更加符合实际的生产需求。

本文研究加工时间不确定的鲁棒一致并行机调度问题，目标函数为调度解的最大完工时间。其中不确定的加工时间用离散场景进行表示。本文在传统的鲁棒调度模型基础上进行拓展，建立了一种两阶段的阈值坏场景集鲁棒优化模型。在第一阶段进行阈值合理取值范围的求解。然后在第二阶段在合理的取值范围内确定不同的阈值作为基准，筛选出未达到基准性能的坏场景，通过优化所有坏场景下的性能与基准性能间的惩罚值来提升系统抵抗坏场景下性能下降这一风险的能力。本文根据该阈值坏场景集鲁棒模型的问题特点设计了一种结合了局部搜索的混合布谷鸟算法，并在局部搜索中设计了一种面向问题的合并坏场景集邻域构造方式。最后通过实验仿真来验证了该邻域构造方式和混合算法的有效性。本文主要做了以下的工作：

1）对确定性一致并行机调度问题和不确定性一致并行机调度问题的研究现状进行了综述。简单介绍了用于一致并行机调度问题的精确算法以及启发式算法，对其中的割平面法和布谷鸟算法进行了详细的概述。同时对传统的场景描述下用于一致并行机调度的鲁棒模型优缺点进行了概述分析。

2）本文针对采用离散场景描述不确定加工时间的一致并行机调度问题提出了阈值和坏场景的概念，并以此为基础建立了一个两阶段的阈值坏场景集惩罚模

型，在第一阶段中先求解阈值的合理取值范围，然后在第二阶段中求解合理区间内不同阈值下的所有坏场景集惩罚值总和，最后提供一系列阈值和不同阈值下求得的坏场景集惩罚模型鲁棒解供决策者进行选择。

3）为了求解该两阶段模型本文设计了一个两阶段求解框架。在第一阶段先对离散场景下的均值场景模型进行求解，得到阈值合理取值范围的下界。第二阶段则是将阈值从下界出发以一定步长进行增长，求解不同阈值下对应的坏场景集惩罚模型。在第一阶段中本文先将离散场景下的一致并行机调度问题转换成单一均值场景下的确定性一致并行机问题，然后采用一种精确的割平面法进行求解，并通过大量的实验仿真证实了该精确算法可以对不同规模的一致并行机算例进行有效的求解。在第二阶段中本文根据问题的特点，设计了一种面向问题的合并坏场景集邻域构造方式，并将采用了该邻域构造方式的局部搜索与布谷鸟算法结合，形成了基于合并坏场景邻域的布谷鸟算法 UNCSA 用于该模型的求解。本文同时采用 UNCSA 和已有的三种智能优化算法进行新模型的求解，通过实验结果证明了本文设计的基于坏场景邻域的布谷鸟算法 UNCSA 在求解第二阶段模型时的有效性和优势。最后通过实验仿真验证了两阶段模型的相关性质。

关键词：鲁棒一致并行机调度；鲁棒优化模型；离散场景；布谷鸟算法；合并场景邻域

ABSTRACT

The production scheduling problem is one of the cores of manufacturing. Excellent production decision-making can make a suitable production plan to ensure the stability and efficiency of production. Parallel machine scheduling problem (PMSP) is a classic type of production scheduling problem, and it is also one of the links connecting different scheduling problems. As the common type of parallel machine scheduling, identical parallel machine scheduling (IPMS) is widely used in various manufacturing fields. Therefore, the research on the identical parallel machine scheduling problem has practical and theoretical value.

At present, the research of identical parallel machines scheduling mainly focuses on the deterministic problem, but the actual production environment is full of various uncertain factors. This makes it difficult for the production plan obtained by solving the traditional deterministic model to meet the production expectations. Therefore, studying the identical parallel machines scheduling problem in the uncertain environment is more able to meet the actual demands.

In this paper, we study the robust identical parallel machine scheduling problem with uncertain processing time, the objective function is the maximum completion time makespan and uncertain processing time is described by discrete scenarios approach. A two-stage robust model is established for identical parallel machine scheduling problems based on a set of bad scenarios which is measured by a threshold performance. The robustness is evaluated by a penalty function on the bad scenarios. According to the problem characteristics of this robust models, this paper designs a hybrid cuckoo algorithm that combines local search strategy, and uses a problem-specific neighborhood construction to improve the efficiency of local search. And the effectiveness of the neighborhood construction and hybrid algorithm is verified through experimental simulation. The main work done in this paper is as follows:

1） We summarized the research status of deterministic identical parallel machine scheduling problem and uncertain identical parallel machine scheduling problem. We briefly introduce the exact algorithms and heuristic algorithms used in the identical parallel machine scheduling problem, and gives a detailed overview of the cutting plane algorithm and the cuckoo search algorithm. At the same time, the advantages and

disadvantages of the traditional robust models for identical parallel machine scheduling problem under scenarios are introduced and analyzed.

2） The concept of threshold and bad scenes is proposed for the identical parallel machine scheduling problem. And a new two-stage threshold-based penalty robust optimization model is established, the reasonable value range of the threshold is solved in the first stage, and then determine a value in the range as the threshold for the next stage. In the second stage, the sum of the penalty values of all bad scenarios under this threshold is used as the robust optimization model to solve, and finally a series of thresholds and the robust solutions of the bad scene set penalty model obtained under different thresholds are provided for decision makers to choose.

3）In order to solve the two-stage model, this paper designs a two-stage algorithm. First, the identical parallel machine scheduling problem in the discrete scenarios is converted into a deterministic identical parallel machine problem in a single mean scenario, and then the cutting plane algorithm is used to solve it. Then in the second stage, according to the characteristics of the problem, a problem-specific neighborhood structure which is constructed by uniting all single-scenario neighborhoods is designed. This constructed neighborhood structure is applied in a hybrid united-neighborhood cuckoo search algorithm (UNCSA), which combined the cuckoo search algorithm with local search. Both UNCSA and three existing algorithms are used to solve the new robust model. The experimental results prove the effectiveness and advantages of the UNCSA proposed in this paper.

Keywords: Robust identical parallel machine scheduling; Robust optimization model; Discrete scenarios; Cuckoo search algorithm; United-scenario neighborhood

目　录

第一章　绪论

1.1 引言

在企业生产制造过程中，生产调度问题是一直需要解决的核心问题之一。通过求解生产调度问题，可以在一定的时间内对有限的生产资源进行合理的分配，制定合适的生产计划来满足特定的性能指标[1]。其本质是提高生产效率，保证生产的稳定高效，优化资源配置，降低生产成本，从而提高企业的经济效益。因此对生产调度问题的研究有着重大的现实意义。

根据生产过程中的工艺约束和设备特征，一般可以将生产调度问题分为单机调度问题（Single Machine Scheduling Problem, SMSP）、并行机调度问题（Parallel Machine Scheduling Problem, PMSP）、流水车间调度问题（Flow Shop Scheduling Problem, FSSP）和作业车间调度问题（Job Shop Scheduling Problem, JSSP）。并行机调度作为经典的生产调度问题，广泛的存在于各种生产制造领域，是联系不同调度问题的纽带之一。而一致并行机调度（Identical Parallel Machine Scheduling, IPMS）又是并行机调度问题中最普遍的一种类型。在一致并行机调度中，所有的机器加工能力相同，这种生产特性符合现实中的多种场景，如芯片制造[2]，螺母加工[3]，轮毂生产[4]等。通过求解一致并行机调度问题可以为优化生产过程，为企业提供可靠的生产加工方案。因此对一致并行机调度问题的研究有着十分重要的理论意义和实用价值。

经典的对于一致并行机调度问题的研究通常都在确定性环境下进行，及各种生产要素都是固定已知的。然而实际的生产环境中充满了各种不确定因素[5,6]，如工人熟练度不同，机器突发故障，现场意外干扰等。此时传统的确定性优化模型与实际的加工模型之间存在一定的差距，求解确定性调度模型得到的方案难以达到生产预期。设计更多用于不确定环境下的优化目标和优化模型，可以为决策者提供更合理的决策权衡，做出更符合实际生产情况的生产计划，充分减少不确定因素造成的影响，因而对不确定一致并行机调度问题的研究近年来也逐渐成为研究的热点。

本论文将重点研究加工时间不确定的一致并行机调度问题。在本章中将首先介绍一致并行机调度问题的研究现状，对用于并行机调度问题的不同求解算法进行阐述。然后介绍并行机调度问题在不确定环境下的研究成果，具体包括不确定参数的建模方法和不确定优化模型等。最后给出本文的内容安排。

1.2 并行机调度问题研究现状

作为经典的生产调度问题之一，并行机调度问题可以看作是联系单机问题和更复杂的多机调度问题的重要纽带，在过去的数十年里，国内外研究学者对并行机调度问题进行了深入的研究，产生了大量的学术文献，为并行机调度的发展打下了坚实的基础。

McNaughton[7]在 1959 年发表的论文是对于并行机最早的研究之一。他研究了带有截止日期的并行机调度问题，并在文中对并行机问题提出了三种性能指标。分别为依赖完工时间的性能（Completion Time Based Performance, CTB），依赖截止日期的性能（Due-Date Based Performance, DDB）和依赖加工流水时间的性能（Flowtime Based Performance, FTB）。虽然其并未在论文中给出具体的求解算法，但在此之后，并行机调度问题开始进入研究者的视野。

关于并行机调度问题的研究在过去的几十年里不断地得到充实和完善。与现实的各种应用场景的联系也逐渐密切。随着求解时目标函数的多样化，求解时约束和规模不断增加，求解的难度也在与日俱增。用于并行机调度问题的求解算法也在不断地进步。从最初的一些简单优化规则，到利用运筹学的一些精确算法，再到融合了不同学科领域技术的启发式算法，算法的求解速度和求解质量与日俱增，能够求解的并行机调度问题的规模也在逐渐提升。发展至今，用于并行机调度的求解算法上已经有了丰硕的研究成果。这些求解算法主要可以分为两大类，精确算法和近似/启发式算法。

精确算法通常建立在数学规划的基础上，利用穷举法，分支定界法，或者割平面法等数学方法进行求解。分支定界算法（Branch And Bound algorithm, B&B）是一类有效的求解离散规划问题的优化算法，其本质是一种隐枚举算法。该算法在求解时将问题的解空间分解成不同的子集，通过计算不同子集的上下界抛弃不符合的解来缩小搜索范围。分支定界算法在并行机调度中有着广泛的应用。Dell

等[8]采用了分支定界算法对目标函数为最大完工时间的一致并行机问题进行了求解，并对问题的上下界进行了优化。实验结果表明该方法可以求解各种规模的算例，求解结果优于一般的动态规划算法。Shim 等[9]在求解可分段加工的一致并行机调度问题时，针对最小化总滞后时间的目标函数计算了新的下界，并以此设计了新的分支定界算法。Haouari 等[10]在分支定界算法中设计了更加紧缩的增强型下界构造方式，并将其用于一致并行机最大完工时间的求解。通过大量的算例证明了该方法的求解效率。除了分支定界法外，Mokotoff[11]设计了一种割平面法。他将一致并行机调度问题用数学规划的形式表示，然后对其进行松弛。通过不断求解松弛模型，根据每次求解结果生成有效割作为新的约束添加到松弛模型中继续求解的方式来求得最优解。该割平面法简单有效，表现出了比分支定界方法更优秀的求解能力。除此之外，Dell 等[12]提出了一种分支定价算法，Chen 等[13]采用了一种列生成算法来进行一致并行机调度问题的求解。

以上提到的都是用于求解一致并行机问题的精确算法。然而一致并行机调度问题是 NP-hard 问题[14]，当问题规模增大时，精确算法的求解时间和求解难度呈指数型增加，很难在合理的时间内获得满意的结果。为了提高求解效率，启发式算法在并行机调度领域的应用逐渐成为研究的重点。根据算法机理的不同，启发式算法可以分为构造启发式算法和迭代启发式算法。构造启发式算法通常基于一些启发式规则进行计算。这类规则问题针对性很强，在求解特定问题时可以简单快速地获得近似最优解。Graham[15]提出了最长加工时间优先规则（longest processing time first, LPT），即各工件按照加工时长依次放到空闲机器上进行加工，并证明在 m 台机器的并行机调度问题下，通过该规则得到的 makespan 与最优性能的性能比不大于 $(\frac{4}{3}-\frac{1}{3m})$。LPT 规则在并行机调度问题中通常用于一些上界的计算，或是用于产生初始解[11,16]，以此来减少求解时间。除了 LPT 规则外，MULTIFIT 算法[17]，COMBINE 算法[18]，LISTFIT 算法[19]等都是应用于并行机调度问题的构造启发式算法。

相比构造启发式算法，迭代启发式算法凭借着通用性强，可以对解空间进行深入搜索等特点越来越受到研究人员的重视。迭代启发式算法往往从一个初始解或由多个初始解组成的初始种群出发，根据一定的机制或者规则对解空间进行搜索。在每一次的迭代中对当前解或当前种群进行更新，在达到终止标准后停止迭

代。根据每次进行迭代的解数量，可以将迭代启发式分为邻域搜索算法和群智能搜索算法。邻域搜索算法是一类串行搜索算法，从单一初始解出发，每次迭代时对当前解产生一批候选解的集合（邻域），从中选取一个邻域解对当前解进行替换更新。常见的邻域搜索算法有模拟退火算法（Simulated Annealing, SA），禁忌搜索算法（Tabu Search, TS），变邻域搜索（Variable Neighborhood Search, VNS）等。SA 是一种模拟物理固体退火过程的算法，利用 Metropolis 准则来防止算法陷入局部最优。Lee 等[16]利用 SA 对一致并行机调度问题进行求解，采用 LPT 准则来产生初始解，通过实验仿真证明了 SA 相比其他构造性启发式算法由更强的求解能力。TS 利用禁忌表来“记忆”一定次数的操作，从而防止发生解的迂回更新，避免算法陷入局部最优。Yamashita 等[20]采用 TS 来优化一致并行机调度问题中的工件平均滞后时间，并通过增加禁忌表长度和设计更有效的解移动方式来提升对解空间的探索能力。邻域构造是邻域搜索算法中的重要步骤，VNS 对同一个当前解采用多种邻域构造产生不同的邻域解，通过增加邻域解产生的多样性来提升每代进行邻域搜索的范围。再从中选出解进行更新。Alharkan 等[21]对最小化 makespan 的一致并行机调度问题采用了五种邻域构造方式来产生邻域解，并分别用随机产生和 LPT 规则两种方式生成初始解。仿真结果表明该改进的变邻域搜索算法比起一般的邻域搜索算法可以获得更好的平均 makespan。

群智能搜索算法以多个解组成的种群为算法的起点，通过模仿自然界中的一些生物行为或者自然规律对解空间进行全面的探索。如模仿生物进化的遗传算法（genetic algorithm, GA），参考鸟群捕食的粒子群算法（Particle Swarm Optimization, PSO），模仿布谷鸟下蛋行为的布谷鸟算法（Cuckoo Search Algorithm, CSA），模拟蚂蚁觅食行为的蚁群算法（Ant Colony Optimization, ACO）等。群智能搜索算法在并行机调度问题中有着各种应用。Liu 等[22]采用了 GA 对一致并行机调度问题进行求解，利用交叉变异等方式对种群进行更新，并通过实验证明求解大规模问题时 GA 比 SA 和 LPT 表现更佳。PSO 算法一开始是用于连续优化问题的求解，Kashan 等[23]在此基础上加入了离散化的步骤来处理离散的调度解，提出了一种离散化后的 PSO 算法（Discrete Particle Swarm Optimization, DPSO）用于求解目标函数为 makespan 的一致并行机调度问题。Neto 等[24]采用蚁群算法对部分生产可进行外包的一致并行机调度问题进行了求解，优化目标是

外包费用和延期损失的花费总和。并提出了三条种群移动规则来保证每个解朝着优化目标函数的方向进行更新。并在每次迭代中加入了局部搜索的步骤来提升求解质量。Laha 等[25]在求解一致并行机调度问题时提出了一种改进布谷鸟算法（Improved Cuckoo Search Algorithm, ICSA），对传统的 CSA 加入了离散化的策略使得 CSA 中用于种群更新的莱维飞行能适用于调度问题。同时还加入了局部更新的步骤来增强算法的搜索能力。实验结果表明布谷鸟算法有着强大的全局搜索能力，相比 PSO, SA，GA 等能获得更好的求解结果。

在并行机调度问题的众多求解方法中，相比精确算法，迭代式启发式算法，或称为智能优化算法，有着更强的求解效率和通用性，在较大规模或复杂目标函数的并行机调度问题的求解中应用广泛。这些智能优化算法有着各自的特点，对解空间有着不同的搜索方式。群智能搜索算法在每次迭代过程中都会对一整个种群进行更新，全局搜索能力较强。而邻域搜索算法则是从当前解出发，构造不同的邻域解进行研究更新，有着较强的局部搜索能力。通过在群智能搜索算法中加入邻域搜索算法或者局部搜索策略的方式来构造混合算法，可以实现不同搜索方式的优势狐不，进一步增强对解空间的搜索能力。目前的研究显示，相比单一的智能优化算法，混合智能优化算法通过结合不同算法的优点，在对一致并行机调度问题的求解中表现得更加出色[46-48]。

如 Wang 等[46]在优化有工艺约束的并行机调度问题时，采用了一种混合了禁忌搜索的遗传算法（Tabu-Genetic Algorithm），用禁忌搜索来增加遗传算法的局部搜索能力。Lee 等[47]在求解一致并行机调度问题时，在遗传算法的基础上加入了局部搜索（GA+LS），通过混合算法的形式来提升求解效率，其中每个个体在每次迭代过程中都会有一定的概率被其邻域解代替。和声搜索算法（Harmony Search, HS）是一种群智能优化算法，是对多个乐师创作歌曲的一种模，有着良好的全局搜索能力，但其在局部搜索能力上依然有改善的空间。Chen 等[48]在和声算法的基础上提出了动态和声算法（Dynamic Harmony Search, DHS）来求解最小化 makespan 的一致并行机调度问题，把整个和声库分为多个子和声库分别进行独立的进化更新。同时在每次迭代过程中加入了 VNS 算法，根据并行机调度问题的特点设计了四种邻域构造方式，利用 VNS 来对每次产生的邻域解进行局部搜索，以此来增强 DHS 的局部搜索能力。Kashan 等[23]在其提出的离散粒子群

算法的基础上加入了局部搜索的环节，形成了混合离散粒子群算法（Hybridized Discrete Particle Swarm Optimization, HDPSO ）。实验结果表明加入局部搜索后的算法明显提高了求解能力，在对大规模一致并行机问题的求解中无论是求解花费的时间还是求解的结果都优于之前的 DPSO。上文提到的布谷鸟算法利用莱维飞行进行每代种群的更新，有着强大的全局搜索能力，但也存在局部搜索能力不强的缺点。Guo 等[59]采用了三种面向一致并行机调度问题的邻域结构，并将采用了这些邻域的局部搜索与布谷鸟算法混合，用于对每代种群中性能表现较差的个体进行更新，将布谷鸟算法引向更好的搜索空间，实现了不同搜索方式的优势互补。而加入了局部搜索的布谷鸟算法也比单一的布谷鸟算法表现出了更好的实验结果。

1.3 不确定性并行机调度问题研究现状

上一小节介绍了近些年来并行机调度领域的丰富研究成果。然而大部分的研究都是在理想的确定性环境下进行的，即各种生产参数如加工时间等都是已知且确定的。然而由于存在工人加工熟练度，机器故障，现场意外的加工干扰等不确定因素[26-28]，传统的确定性模型和实际加工模型之间存在着一定差距，此时按照求解确定性模型得到的方案进行加工制造难以达到生产预期。因此对不确定并行机调度问题进行研究有着重要的学术意义和实际价值。如何处理并行机调度中的不确定参数，建立合适的不确定模型并设计不确定环境下的高效求解算法逐渐成为了研究的热门。

1.3.1 不确定参数建模方式

对于如何处理不确定参数，目前主要有随机分析法，模糊分析法和场景法三种方式。在随机分析法中，不确定参数用随机变量表示，相应的随机变量利用概率分布模型进行描述。在随机分析法中通常对实际生产中的大量历史数据进行反复分析，利用得到的统计规律来确定其概率分布模型。常见的模型有指数分布，正态分布等。如 Alimoradi 等[29]在优化一致并行机调度的总流水时间时，将不确定的工件加工时间用满足正态分布的随机变量表示，并设计了一种分支定界算法进行求解。Ranjbar 等[30]提出了两种分支定界算法来求解最大化顾客满意度的一

致并行机调度问题，其中每个工件的加工时间满足正态分布。Cai 等[31]研究了每台机器都会有随机故障发生的并行机调度问题，优化目标是最小化提前完工造成的花费和总滞后工件数。其中的交货期满足指数分布。

在有些情况下通过对历史数据的分析只能获得不确定参数的一些近似值，很难得到不确定参数服从的具体概率分布。此时可以使用模糊分析法来进行不确定参数的处理。同随机分析法不同，在模糊分析法中，不确定参数不再满足特定的概率分布模型，而是引入了模糊数的概念，利用模糊数和对应的模糊规则来对不确定参数进行描述和预测[32]。Balin[33]用三角模糊数来表示不确定并行机调度中的加工时间，采用 GA 来优化最大完工时间的目标。并通过测试不同模糊集下的问题，将 GA 求得的结果与 LPT 方法获得的结果进行了比较来体现 GA 算法的有效性和稳定性。Yeh 等[34]在优化不确定并行机调度的最大完工时间时，用梯形模糊数来表示不确定的加工时间，并同时采用了 SA 和 GA 两种智能优化算法来进行求解对比。Behnamian[35]用钟形模糊数来描述不确定的加工时间，并采用了面积中心法来进行解模糊。同时设计了一种离散粒子群算法，在粒子群算法中加入了遗传算法的交叉变异操作，并用该改进的算法求解较大规模的模糊一致并行机调度问题。

第三种是场景法，其中的每一个场景代表一种可能的生产环境。通过将各种不确定因素包含在若干场景中实现建模。随机分析法和模糊分析法需要大量的数据和经验作为基础来建立随机概率分布或模糊隶属度函数。然而在现实的生产环境中，经常会出现诸如新工艺，新生产线等历史数据不足的生产场景[36,37]，此时采用随机分析法或模糊分析法进行建模有一定的困难。相比之下，场景法不需要历史数据的积累，适用于在不确定因素的发生概率未知或历史信息不足的情况下描述不确定参数。场景法可以分为区间场景（Interval Scenario）和离散场景（Discrete Scenario）两种形式[38]。区间场景下的不确定参数可在一个上下界内任意取值，本质上可以看作有无限多个场景。而离散场景则是通过有限数量的场景来描述不确定性，一个场景代表一组可能的参数离散值，各场景之间相互独立互不影响。

一般不确定参数的概率分布函数或者隶属度函数不容易直接获得，而场景的构建相对容易。综合考虑三种建模方式的特点，本文将选用离散场景来描述一致

并行机调度问题中的不确定参数。接下来将对基于场景的不确定模型进行介绍。

1.3.2 基于场景的不确定调度模型

与确定性问题不同，在不确定环境下需要根据系统的不确定性建立新的调度模型。在利用场景法进行不确定参数建模的问题中，Li 等[39]将基于场景的不确定模型分为两类，基于场景的随机优化模型和基于场景的鲁棒优化模型。两种模型的优化目标不同，体现了不同的决策偏向。

基于场景的随机优化模型将整体期望性能作为优化目标[40]，考虑优化整个调度解在所有可能场景下性能的平均水平。随机优化模型的优化目标代表了统计意义上对优良性能的追求，体现了决策的积极性。在一致并行机调度问题中有一定的应用。如 Liu 等[41]研究了离散场景下的一致并行机调度问题，不确定加工时间用多个离散场景的集合表示，优化目标是总完工时间在所有场景下的期望值。并且分别采用抽样平均近似法（Sample Average Approximation, SAA）和 GA 来对不同规模的算例进行求解。

随机优化模型缺乏对某些场景下表现出较差性能这一可能的抵抗能力，抗风险性较弱。而基于场景的鲁棒优化模型则是以增加系统抵抗不确定性干扰的能力为优化目标，体现了决策者的风险厌恶偏向。鲁棒模型中比较常见的有 min-max 模型（最坏场景模型）和 min-max regret 模型（最大后悔场景模型），通过优化所有场景下的最坏性能或者最大遗憾值来提升系统的抗风险能力。如 Xu 等[42]对区间场景下的一致并行机最大 makespan 的遗憾值进行优化，分别采用了松弛迭代算法和 SA 来求解不同规模的问题。Drawl 等[43]对区间加工时间下的一致并行机调度问题进行了研究，优化目标是最大完工时间的最大遗憾值。Wang 等[44]研究了工件可外包的一致并行机调度问题，目标函数是外包消耗花费和非外包加工花费的生产总成本。他们同时采用了离散场景和区间场景来描述不确定的加工时间，利用 2-近似算法（2-Approximation Algorithm）分别对区间场景下生产总成本的最大遗憾值和离散场景下的最大生产总成本问题进行了求解。但这类传统的鲁棒优化模型也存在一定的不足。因为过度关注解在单一最坏场景下的表现，往往忽视了该解在其他场景下的性能表现，导致最终获得求解结果往往过于保守，缺乏对积极性的追求。针对该不足目前也出现了一些新的鲁棒模型，如 Wang 等[45]以

一个性能值为阈值，以此筛选出性能表现劣于阈值的坏场景，通过综合考虑所有坏场景下的性能而不是单一的最坏场景性能来减弱最终调度解的保守性，在鲁棒性和积极性之间实现了更好的均衡。然而该模型目前只在鲁棒作业车间调度问题中进行应用，尚未在鲁棒并行机调度中进行具体的使用，本文希望在此基础上进行尝试和突破。

以上提到的随机分析法、模糊分析法和场景法都是并行机调度中处理不确定参数的有效方法。然而随机分析法和模糊分析法都需要大量的历史数据，其概率分布函数或者模糊隶属度函数一般不太容易获得。相比之下场景法的描述方式更加准确且容易实现。而在不确定环境下，获得抵御风险能力的结果体现了很多决策者的保守追求，因此基于场景的鲁棒优化模型逐渐成为了研究的热门。综上考虑，本文将采用离散场景来进行描述一致并行机调度问题中的不确定加工时间，并以此建立合适的鲁棒模型进行求解。同时根据第二节中不同算法在并行机调度问题中的研究现状，综合布谷鸟算法在各群智能算法中的优异表现，本文将在布谷鸟算法的基础上设计混合算法进行求解。采用布谷鸟算法加入局部搜索的方式来对本文研究的一致并行机鲁棒调度问题进行求解。同时本文还将根据研究的问题模型特点设计面向问题的邻域构造方式，让局部搜索更具效率。混合算法的细节将在第三章中进行详细的描述。

1.4 本论文的研究内容及章节安排

本论文针对加工时间不确定的一致并行机鲁棒调度问题开展研究，目标函数是最大完工时间 makespan。首先采用离散场景来描述不确定加工时间，然后求解一致并行机的坏场景集鲁棒调度模型。这是在传统鲁棒优化模型上进行拓展的一种优化模型，通过关注更多坏场景下的性能来降低最终鲁棒解的保守性，实现对鲁棒性和积极性之间的均衡追求。本文设计求解了一个两阶段阈值坏场景集优化模型，并设计了一个两阶段算法进行求解。在第一阶段对已有的期望场景模型进行转换然后采用割平面法求解阈值，然后结合一致并行机调度问题和该模型的特点，设计了一种面向问题的邻域构造方式，并将采用该邻域方式的局部搜索与布谷鸟算法结合用于第二阶段阈值坏场景集惩罚模型的求解。最后通过仿真实验来证明所设计的算法的有效性以及该两阶段鲁棒优化模型的性质。

基于本文所要研究的内容，各章节安排如下：

第一章绪论，首先介绍了确定性并行机调度问题的研究现状，然后介绍了不确定并行机调度问题的研究现状，包括一些常见的求解算法。接着介绍了处理不确定参数的不同方法以及常见的基于场景的不确定调度模型，引出本文的研究内容。

第二章中详细介绍了一致并行机调度问题，给出了确定性一致并行机调度问题和场景集下不确定一致并行机优化常见优化模型的数学规划形式，并给出了本文要求解的两阶段阈值坏场景集模型框架。然后介绍了求解一致并行机问题的常规算法，包括割平面法和布谷鸟算法。并对一致并行机调度中常用的邻域构造方式进行了详细的阐述。

第三章是对加工时间不确定的一致并行机调度问题进行研究，采用离散场景表示不确定加工时间，在阈值确定的情况下对离散场景下的一致并行机阈值坏场景集惩罚模型进行求解。本章中设计了一种面向模型的合并场景邻域方式，并将采用了该邻域的局部搜索加入到布谷鸟算法中形成一种混合布谷鸟算法，然后将混合算法与其他三种已有的算法一起用于该模型的求解，通过仿真测试结果证明该混合布谷鸟算法的有效性。

第四章在第三章的基础上，对离散场景下一致并行机调度问题的两阶段阈值坏场景集模型进行求解。针对该模型本文设计了一种两阶段求解算法，在第一阶段采用精确算法中的割平面法来求解阈值，在第二阶段采用第三章设计的混合布谷鸟算法进行求解。同时实验仿真验证了该两阶段算法求解的有效性，并同时对该两阶段模型的性质特点进行了验证。

第五章对本文进行了总结和展望。

第二章　一致并行机调度问题及其求解算法

2.1 引言

并行机调度问题属于典型的生产调度问题，是联系单机调度问题和其余复杂调度问题的纽带[49]。一致并行机调度是并行机调度中最普遍的加工类型，涉及多台加工能力相同的机器，在实际生产中有着广泛的应用。其在确定性环境下的调度模型和不确定环境下的鲁棒模型都是研究的热门。用于一致并行机问题的求解算法多种多样，可以满足不同精度和规模的需求。在精确算法中，割平面法凭借高效易于操作的特点被广泛地使用。而在启发式算法中，布谷鸟算法作为一种较为新颖的群搜索算法，有着强大的全局搜索能力，经常通过混合局部搜索，增加对邻域解的搜索来提升求解效率，被应用于各种调度问题包括一致并行机调度中。

基于以上内容，本章的内容安排如下：2.2 节介绍确定性一致并行机问题及其数学模型。2.3 节和 2.4 节是对不确定环境下基于场景的一致并行机优化模型介绍。2.5 节是对求解一致并行机的割平面法的介绍，2.6 节是对标准布谷鸟算法的介绍，包括了算法原理和基本步骤。2.7 节是对一致并行机调度中的局部搜索以及用于一致并行机邻域构造方式的介绍。2.8 节是对本章内容的总结。

2.2 确定性一致并行机调度问题

并行机调度问题涉及 n 个工件，　m 台机器 $(n>m)$　，J_i 表示第 i 个工件，$J=\{J_i \mid i=1,2,...,n\}$ 为所有工件的集合；M_j 表示第 j 台机器，$M=\{M_j \mid j=1,2,...,m\}$ 为所有机器的集合。p_{ij} 表示 J_i 在 M_j 上的加工时间。同时在并行机调度问题中，还有以下的约束：

- 任意一个工件可在任意一台机器上进行加工，工件之间相互独立，没有优先级。
- 一台机器同一时刻只能加工一个工件，一旦开始加工无法中途停止。
- 一个工件只需加工一次。
- 无特殊情况，所有的工件和机器在 0 时刻都是准备就绪的。

根据加工机器的属性，并行机调度问题可以分为以下三类：一致并行机调度问题，异速并行机调度问题和无关并行机调度问题[50,51]。

一致并行机调度问题中的所有机器没有差别，同一个工件在不同机器上的加工时间完全相同，加工时间只与工件有关。即对于 $J_i,(i=1,2,..,n)$，其在不同机器上的加工时间可统一记作 p_i，有 $p_{i1}=p_{i2}=...=p_{im}=p_i$。

异速并行机调度中的每台机器各自有一个速度系数。对于机器 $M_j(j=1,2,...,m)$，其速度系数记作 s_j。每个工件 $J_i(i=1,2,...,n)$ 有一个基本加工时间 p_i，工件在不同机器上的加工时间满足 $p_{ij}/s_j=p_{ik}/s_k=p_i,(j,k=1,2,...,m)$。

无关并行机中的工件加工时间与机器和工件均相关。同一个工件在不同机器上加工，同一台机器加工不同的工件的时间可能都不相同。即不同的 $p_{ij},(i=1,2,...,n;\ j=1,2,...,m)$ 之间并无特殊的规律。

本文中研究的一致并行机调度问题，目标函数均为最大完工时间 makespan。用 SX 表示所有调度解的集合，$X=[x_{ij}]_{n\times m}\in SX$ 为一个具体的可行调度解，如果选择在 M_j 上加工 J_i，则 $x_{ij}=1$，否则 $x_{ij}=0$。$C_j(X)$ 表示机器 M_j 在调度方案 X 下的完工时间，为在 M_j 上加工的所有工件的加工时间之和：

$$C_j(X)=\sum_{i=1}^{n}p_i x_{ij}\quad j=1,...,m \tag{2.1}$$

一致并行机调度中所有机器中最长的完工时间即为最大完工时间，记作 $C(X)$：

$$C(X)=\max_j\{C_j(X)\}=\max_j\{\sum_{i=1}^{n}p_i x_{ij}\}\quad j=1,...,m \tag{2.2}$$

当所有工件的加工时间 $p_i(i=1,2,...,n)$ 都为确定的数值时，这样的问题被成为确定性一致并行机调度问题。

以 makespan 为目标函数的一致并行机问题（Identical Parallel Machine Scheduling，IPMS），用三元法表示为 $P_m\|C_{\max}$，该问题可表示为如下的数学规划模型形式：

（IPMS）

$$\min\quad y \tag{2.3}$$

$$s.t \ \sum_{j=1}^{m} x_{ij} = 1, \ 1 \le i \le n, \tag{2.4}$$

$$y - \sum_{i=1}^{n} p_i x_{ij} \ge 0, \ 1 \le j \le m, \tag{2.5}$$

$$x_{ij} \in \{0,1\} \tag{2.6}$$

2.3 基于场景的不确定一致并行机随机优化模型

在实际情况中，各工件的加工时间往往是无法确定的。本文采用 1.3.1 节介绍过的离散场景来描述不确定的加工时间，研究场景集下的一致并行机调度问题（Scenario Identical Parallel Machine Scheduling, SIPMS）。本文中用 Λ 表示所有离散场景的集合，$|\Lambda|$ 为场景集 Λ 中的场景总数。$\lambda=\{p_1^{\lambda}, p_2^{\lambda}, ..., p_n^{\lambda}\} \in \Lambda$ 表示一个具体的场景，为一组可能的加工时间取值。$C_j(\mathrm{X},\lambda)$ 表示调度解 X 中的机器 M_j 在场景 λ 下的完工时间，$C(\mathrm{X},\lambda)$ 表示在场景 λ 下调度解 X 的 makespan：

$$C(\mathrm{X},\lambda) = \max_j \{C_j(\mathrm{X},\lambda)\} = \max_j \{\sum_{i=1}^{n} p_i^{\lambda} x_{ij}\} \ j = 1, ..., m \tag{2.7}$$

可以看出，在给定场景 λ 后，以式(2.7)为目标函数的 SIPMS 问题可以看作是上节提到的表达为 $P_m \| C_{\max}$ 的确定性一致并行机调度问题。

场景下的不确定一致并行机调度问题存在不同的优化模型，可以分为基于场景的随机优化模型和基于场景的鲁棒优化模型[39]，体现了不同的决策偏向。

随机优化模型通常以所有场景下的期望性能作为优化指标，因此也称为期望模型[52]。给每个场景 λ 赋予发生概率 $\gamma(\lambda)$，用 $EC(\mathrm{X})$ 表示调度解 X 在 Λ 中的所有可能场景下的期望性能指标（Expected-performance Criterion），以 $EC(\mathrm{X})$ 作为优化指标的期望模型（Expected-performance Criterion Model, ECM）的表达式如下所示：

（ECM）

$$\min_{\mathrm{X} \in SX} EC(\mathrm{X}) = \min_{\mathrm{X} \in SX} \sum_{\lambda \in \Lambda} \gamma(\lambda) C(\mathrm{X},\lambda) \tag{2.8}$$

而在本文研究的离散场景中，所有的场景均以“纯”场景的方式出现，即不同场景之间没有发生概率的差别，或者说不同场景出现的概率相同。此时优化的期望性能表现为所有场景下的均值性能。用 $MC(\mathrm{X})$ 表示调度解 X 在 Λ 中的所有

可能场景下的均值性能指标（the Mean-performance Criterion），以 $MC(\mathrm{X})$ 作为优化指标的均值模型（the Mean-performance Criterion Model, MCM）的表达式如下所示：

（MCM）

$$\min_{\mathrm{X}\in SX} MC(\mathrm{X}) = \min_{\mathrm{X}\in SX} 1/|\Lambda| \sum_{\lambda\in\Lambda} C(\mathrm{X},\lambda) \tag{2.9}$$

2.4 基于场景的不确定一致并行机鲁棒优化模型

上节提到的随机优化模型将所有场景下的期望性能作为优化目标，体现了偏积极性的决策偏向。而基于场景的鲁棒优化模型则是以增加整个系统抵抗不确定性干扰的能力为优化目标，把抗风险作为基本的决策偏向。

目前对于一致并行机鲁棒优化模型的研究以 min-max 模型（最坏场景模型）和 min-max regret 模型（最大后悔场景模型）为主[53,54]。

2.4.1 最坏场景模型

对于一个可行调度解 X，其在场景集 Λ 中表现性能最差的场景被称为 X 的最坏场景（the Worst-case Scenario），记作 $\lambda^{w}(\mathrm{X})$：

$$\lambda^{w}(\mathrm{X}) = \arg\max_{\lambda\in\Lambda} C(\mathrm{X},\lambda) \tag{2.10}$$

$WC(\mathrm{X})$ 表示表示调度解 X 在 Λ 中的所有可能场景下最坏的性能（the Worst-scenario performance Criterion）：

$$WC(\mathrm{X}) = \max_{\lambda\in\Lambda} C(\mathrm{X},\lambda) = C(\mathrm{X},\lambda^{w}) \tag{2.11}$$

以 $WC(\mathrm{X})$ 作为鲁棒优化指标的模型，即为最坏场景模型（the Worst-scenario performance Criterion Model，WCM）：

（WCM）

$$\min_{\mathrm{X}\in\mathrm{SX}} \max_{\lambda\in\Lambda} C(\mathrm{X},\lambda) \tag{2.12}$$

WCM 是最传统的鲁棒调度模型之一，通过优化最坏场景下的性能来提升调度解性能下降的风险，体现了决策者的悲观偏向。因为 WCM 只关注单一最坏场景下的性能，忽视了当前解在其他场景下的表现，因此最终得到的鲁棒解过于

保守。

2.4.2 最大后悔场景模型

Kouvelis 等[38]在最坏场景模型的基础上提出了最大后悔场景模型，是一种基于偏差鲁棒指标的鲁棒优化模型。

用 X_{λ}^{*} 表示场景 λ 下的最优调度解。$C(X_{\lambda}^{*},\lambda)$ 为场景 λ 下的最优性能。则对于一个可行调度解 X，其在 λ 下的表现性能与该场景下的最优性能的偏差即为 X 在 λ 下的后悔值（regret），记作 $R(X,\lambda)$：

$$R(X,\lambda)=C(X,\lambda)-C(X_{\lambda}^{*},\lambda) \tag{2.13}$$

而 X 在所有场景下最大的后悔值 $WR(X)$ 被称为 X 的最大后悔值（the Worst-case Regret）：

$$WR(X)=\max_{\lambda\in\Lambda}(C(X,\lambda)-C(X_{\lambda}^{*},\lambda)) \tag{2.14}$$

以 $WR(X)$ 作为鲁棒优化指标的模型，即为最大后悔模型（the Worst-case Regret Model，WRM）：

（WRM）

$$\min_{X\in SX}\max_{\lambda\in\Lambda}(C(X,\lambda)-C(X_{\lambda}^{*},\lambda)) \tag{2.15}$$

相比 WCM，WRM 用与场景下最优性能的相对偏差代替了性能的绝对差值，包含了各场景下最优性能的追求，通过减少后悔值的方式来降低最终解的鲁棒性，但 WRM 依然只关注后悔值最大的单一场景。同时 WRM 的求解需要计算每个场景下的最优性能，对于在确定性环境下就为 NP-hard 的一致并行机调度问题来说无疑增加了求解难度。

2.4.3 阈值坏场景集惩罚模型

前两小节中对传统的鲁棒模型 WCM 和 WRM 进行了介绍，这些模型通过关注单一最坏场景下的性能或者最大遗憾值来获得具有抗风险能力的鲁棒解。Wang 等[45]在此基础上关注更多性能表现较差的场景，以这些场景下的性能为鲁棒衡量指标，提出了阈值坏场景集惩罚模型。

根据 Wang 等[45]的研究内容，首先给出以下的定义：

定义 **2.1** 在用离散场景表示不确定加工时间的一致并行机调度问题中，同一个可行解 X 在不同的场景 $\lambda, \lambda \in \Lambda$ 下表现出不同的性能。给定一个阈值性能 T，令 X 的表现性能劣于 T 的场景被称作 X 的一个坏场景。X 所有坏场景的集合被称为 X 的阈值坏场景集（Threshold Bad-scenario Set, TBS），记作 $\Lambda_T(\mathrm{X})$：

$$\Lambda_T(\mathrm{X}) = \{\lambda \mid C(\mathrm{X}, \lambda) \geq T, \lambda \in \Lambda\} \tag{2.16}$$

从 TBS 的定义可以看出，对于一个可行解 X，其最坏场景 $\lambda^w(\mathrm{X})$ 也包含在 $\Lambda_T(\mathrm{X})$ 中。

将每个坏场景 $\lambda, \lambda \in \Lambda_T(\mathrm{X})$ 下的性能与阈值 T 的差值的平方项作为解 X 在坏场景下未达到标准的惩罚值。将 X 在所有坏场景中的惩罚值相加，定义阈值坏场景集惩罚值 $PT(\mathrm{X})$：

$$PT(\mathrm{X}) = \sum\nolimits_{\lambda \in \Lambda_T(\mathrm{X})} \left[C(\mathrm{X}, \lambda) - T\right]^2 \tag{2.17}$$

以优化 $PT(\mathrm{X})$ 为鲁棒指标，定义一个阈值坏场景集模型 PTM（PT Model）：（PTM）

$$\min\nolimits_{\mathrm{X} \in SX} PT(\mathrm{X}) \tag{2.18}$$

相比 WCM 模型，PTM 模型关注包括最坏场景在内的更多坏场景，因此得到的鲁棒解的保守性大为降低。同时又不需要求解每个场景下的最优性能，相比于 WRM 模型，降低了求解的难度。

2.4.4 两阶段阈值坏场景集模型

上一节中提到了阈值坏场景集惩罚模型，通过优化所有坏场景集下的惩罚值来抑制坏场景下的性能恶化，表达了决策者抵御未达到标准的性能的抗风险偏向。但对于 PTM 存在以下的假设和前提：阈值 T 是已知的，并且基于给定的阈值 T 能够建立有效的 PTM 模型。

从坏场景的定义式(2.16)和 PTM 模型的鲁棒优化性能 $PT(\mathrm{X})$ 的表达式(2.17)可以看出，对于一个解，阈值 T 作为一种可能的性能指标，承担着衡量坏场景 TBS 的作用。同时基于不同的 TBS 会建立完全不同的 PTM 模型。在场景集 Λ 给定的情况下增加 T 值，意味着对于每一个可行调度解 X，其则对应的坏场景集 $\Lambda_T(\mathrm{X})$ 中坏场景的数目也会减少。当阈值 T 增加到一定程度，会出现对于部分可行解，

其在所有场景下的性能表现都优于T，即$\Lambda_T(\mathrm{X})$为空的情况。此时这些解对应的坏场景集惩罚值$PT(\mathrm{X})$均为零，即无法利用 PTM 模型来判断这些解在不确定环境下的鲁棒性好坏。这种情况下，根据该阈值T建立的 PTM 模型是无效的，无法用于对解的筛选决策。

因此在阈值T没有事先给定的情况下，需要先在合理的范围内确定一个阈值，然后基于该阈值建立 PTM 模型进行求解，以此来保证 PTM 模型的有效性。

用$|\Lambda_T(\mathrm{X})|$表示可行解X在阈值T的坏场景数量，用$\mathrm{PTM}|T$进一步表示阈值T下的 PTM 模型，给出如下的定义：

定义 2.2 对于$\forall \mathrm{X} \in SX$，$|\Lambda_T(\mathrm{X})|>0$，则称给定的阈值$T$为合理的阈值。

定义 2.3 在给定合理阈值T下的$\mathrm{PTM}|T$被称为有效的阈值坏场景集惩罚模型。

Wang 等[81]对阈值T进行了研究，证明为了保证$\mathrm{PTM}|T$的有效性，合理的阈值T应该位于每个可行解$\mathrm{X} \in SX$在每个场景$\lambda \in \Lambda$下能实现的所有可能性能的公共区间上。而通过求解最坏场景模型 WCM 下获得的WC^*反应了所有场景中最优的最坏场景性能，即WC^*为这一公共区间的上限。如果选择给定的阈值$T \geq WC^*$，则会出现一些精英解不存在坏场景的现象。即WC^*是阈值T合理取值范围的一个上界。

而根据 Daniels 等[74]和 Wang 等[45]的建议，为了保证求解 PTM 模型最终得到的解能够对抗坏场景中性能下降的风险，阈值T的取值不应小于所有场景下的最优期望性能EC^*，这一性能在“纯”场景背景下表现为所有场景下的最优均值性能MC^*。即MC^*为阈值T合理取值范围的一个下界。综上所述，T的合理取值应该为$EC^* \leq T \leq WC^*$，在本文研究的“纯”场景下也可以表示为$MC^* \leq T \leq WC^*$。

定义 2.4 当T的取值区间为$[EC^*, WC^*]$时，对应的$\mathrm{PTM}|T$为有效的阈值坏场景集惩罚模型，在“纯”场景下这一区间也可以记作$[MC^*, WC^*]$。

可以发现区间$[MC^*, WC^*]$提供了无数个合理的阈值T，对应无数个$\mathrm{PTM}|T$模型。通过求解这些不同的$\mathrm{PTM}|T$模型得到不同的 PT 鲁棒解。不同阈值和其对应的 PT 惩罚值组成的$(T, PT(\mathrm{X}^b, T))$实现了系统鲁棒性和最优性间的不同折中，可以为决策者提供不同决策偏好的选择。

进一步分析PTM|T模型，可以发现该模型的鲁棒性和优化性之间存在着如下的对立统一关系：惩罚值PT反映了系统的鲁棒性追求，较小的PT有益于改善系统整体的鲁棒性。而阈值T作为坏场景性能的衡量指标，反映了系统对优化性的追求。较小的阈值意味着对整体期望性能更高的追求，较小的T值有助于PTM|T模型抑制更多坏场景下的性能，有益于提升系统整体的优化性。对于一个可行解X，如果增加PTM|T模型的阈值T，会使坏场景数$|\Lambda_T(\mathrm{X})|$减少，从而降低$PT(\mathrm{X})$，反之亦然。即对于PTM|T模型来说，鲁棒性和优化性这两个目标是相互矛盾的。

根据PTM|T的这一特点以及定义 2.3 和定义 2.4，Wang 等[81]将PT和T同时作为优化目标，建立了一个双目标 PTM 模型框架（PTM Framework, PTMF）：

（PTMF）

$$(\mathrm{PTM}|T) \qquad \min PT(\mathrm{X}) = \sum\nolimits_{\lambda\in\Lambda_T(\mathrm{X})} \left[C(\mathrm{X},\lambda) - T\right]^2 \tag{2.19}$$

$$\min T \tag{2.20}$$

$$s.t.\ MC^* \le T \le WC^* \tag{2.21}$$

$$(\mathrm{MCM}) \qquad MC^* = \min_{\mathrm{X}\in SX} 1\Big/\left|\Lambda\right| \sum\nolimits_{\lambda\in\Lambda} C(\mathrm{X},\lambda) \tag{2.22}$$

$$(\mathrm{WCM}) \qquad WC^* = \min_{\mathrm{X}\in SX} \max_{\lambda\in\Lambda} C(\mathrm{X},\lambda) \tag{2.23}$$

该双目标模型包含了PTM|T，MCM，WCM 三个鲁棒模型，其中的PTM|T为求解的主模型，MCM 和 WCM 为用于确定合理阈值边界的辅助模型。通过求解 PTMF，可以得到不同的阈值T和该阈值下的PTM|T对应的$PT(\mathrm{X}^b,T)$。这一系列$(T,PT(\mathrm{X}^b,T))$可以提供给不同偏好的决策者制定不同生产计划的决策选择。

相比其他的双目标优化问题，PTMF 的两个优化目标之前存在着一定的求解先后顺序，需要先确定T值再求解该阈值下的PTM|T模型，因此这两者之间虽然互相冲突但是并不侵犯。基于这一特点，可以将 PTMF 模型重新表示成一个两阶段阈值坏场景集模型（Two-Stage PT Model, TSPTM）：

（TSPTM）

第一阶段

$$(\mathrm{MCM}) \qquad MC^* = \min_{\mathrm{X}\in SX} 1\Big/\left|\Lambda\right| \sum\nolimits_{\lambda\in\Lambda} C(\mathrm{X},\lambda) \tag{2.24}$$

$$(\text{WCM}) \qquad WC^* = \min_{\text{X}\in SX} \max_{\lambda\in\Lambda} C(\text{X},\lambda) \tag{2.25}$$

$$T \in [MC^*, WC^*] \tag{2.26}$$

第二阶段

$$(\text{PTM}\,|\,T) \qquad \min PT(\text{X}) = \sum\nolimits_{\lambda\in\Lambda_T(\text{X})} \left[C(\text{X},\lambda) - T\right]^2 \tag{2.27}$$

在 TSPTM 的第一阶段，先求解 MCM 模型和 WCM 模型，得到场景集 Λ 下的 MC^* 与 WC^*。在第二阶段对不同阈值 T 下的 $\text{PTM}\,|\,T$ 进行求解，得到一系列 $PT(\text{X}^b, T)$。

相比单一的 MCM 模型和 WCM 模型，TSPTM 模型兼具对系统鲁棒性和优化性的考量，但尚未在一致并行机调度问题中应用。本文将应用该模型在一致并行机鲁棒调度问题中。由于 MCM 模型，WCM 模型和 $\text{PTM}\,|\,T$ 都是 NP-hard 问题，因此整个 TSPTM 模型的求解具有一定的难度。本文将在后续章节中对其作进一步的处理，设计合适的求解框架进行求解。

2.5 求解一致并行机调度问题的割平面法

Mokotoff[11]在 2004 年提出了一种求解 $P_m \,||\, C_{\max}$ 的割平面法。同其他求解确定性一致并行机调度的精确算法如分支定界法相比，该方法有着更高的求解效率，并可与其他算法结合用于求解更复杂的目标函数。如 Xu 等[42]在求解区间场景下的一致并行机最大后悔模型时就采用了该割平面能法来确定每个极点场景下的最优 makespan。

2.5.1 算法简介

该割平面法是先确定目标函数的上下界限，令目标函数 y 等于当以前下界并逐步增加下界的值。在给定 y 的情况下对原始的 IPMS 模型进行松弛求解，判断所得结果是否为整数解。如果不是整数解，则根据求解结果往松弛模型中增加有效不等式（Valid Inequality）。通过有效不等式对松弛可行域进行切割，通过缩减可行域的范围形成让求解结果向最终的整数解不断接近。

根据 2.2 节中的 IPMS 的数学规划模型，IPMS 问题的解 (x, y) 可行域 F 为：

$$x \in \{0,1\}^{m \times n}, \tag{2.28}$$

$$y \in R^{+}, \tag{2.29}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in \{1,...,\mathrm{n}\} \tag{2.30}$$

$$y - \sum_{i=1}^{n} p_i x_{ij} \geq 0 \quad \forall j \in \{1,...,m\} \tag{2.31}$$

通过将式(2.28)这一约束条件松弛为$x \in [0,1]^{m \times n}$，得到松弛可行域P：

$$x \in [0,1]^{m \times n}, \tag{2.32}$$

$$y \in R^{+}, \tag{2.33}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in \{1,...,\mathrm{n}\} \tag{2.34}$$

$$y - \sum_{i=1}^{n} p_i x_{ij} \geq 0 \quad \forall j \in \{1,...,m\} \tag{2.35}$$

则 IPMS 模型的表示形式从最初的$\min\{y:(x,y) \in F\}$变为了$\min\{y:(x,y) \in P, Ax + Dy \leq b\}$，其中的$Ax + Dy \leq b$即为有效不等式。每次求解过后通过增加下界的值和加入有效不等式的方式对松弛可行域进行切割，直到得到最终的整数解。

极少情况下会遇到无法增加有效不等式但还未求得最终解，此时根据当前的下界利用分支定界法[8]继续进行求解。

2.5.2 有效不等式的计算

Mokotoff[11]对有效不等式给出了具体证明过程，本文中将不对证明过程做详细介绍，只介绍有效不等式的产生方法。

设求解某次的 IPMS 松弛问题得到的非整数解为(x_0, y_0)，$S_j = \{i \in N \mid x_{ij}^0 > 0\}$表示在机器$M_j$上的加工工件集合。将这些工件的加工时间相加，用$\Delta j = \sum_{i \in S_i} p_i - y^0$表示在$M_j$上的额外时间消耗。用$S_j' = \{k \in S_j \mid p_k > \Delta j\}$表示$S_j$中加工时间大于$\Delta j$的工件集合。

对于每台机器M_j，当$\Delta j > 0$时，可以增加如下的有效不等式作为约束：

$$\sum_{i \in S_j} p_i x_{ij} \leq y^0 - \sum_{i \in S_j'} (p_i - \Delta j)^{+} (1 - x_{ij}) \tag{2.36}$$

下面以一个 6 工件 2 机器的一致并行机调度问题为例来展示有效不等式的产生过程，各工件的加工时间 $p_1 \sim p_6$ 为 $\{5,\ 5,\ 3,\ 3,\ 1,\ 1\}$。在 $y^0 = 9$ 时利用 CPLEX 在松弛可行域下对 IPMS 的线性规划模型进行求解，求得的结果如表 2.1 所示：

表 2.1　IPMS 松弛问题示例求解结果

$i \setminus j$	M_1	M_2
J_1	1	0
J_2	0	1
J_3	0.33	0.67
J_4	1	0
J_5	0	1
J_6	0	1

根据求解结果，确定各机器上的加工工件，$S_1 = \{1,\ 3,\ 4\}$, $S_2 = \{2,\ 3,\ 5,\ 6\}$，$\Delta 1 = p_1 + p_3 + p_4 - 9 = 2$, $\Delta 2 = p_2 + p_3 + p_5 + p_6 - 9 = 1$，根据额外时间消耗可以得到 $S_1' = \{1,\ 3,\ 4\}$, $S_2' = \{2,\ 3\}$。由此根据表 2.1 的结果可以确定如下两个有效不等式：

$$5x_{11} + 3x_{31} + 3x_{41} \le 9 - [(5-2)(1-x_{11}) + (3-2)(1-x_{31}) + (3-2)(1-x_{41})] \quad (2.37)$$

$$5x_{22} + 3x_{32} + 1x_{52} + 1x_{62} \le 9 - [(5-1)(1-x_{22}) + (3-1)(1-x_{32})] \quad (2.38)$$

2.5.3　算法步骤

利用割平面法求解 IPMS 分为以下几个主要的步骤：

Step1：将所有的工件按照加工时间的降序进行编号，使得 $p_1 \ge p_2 \ge ... \ge p_n$，这是为了用于接下来的目标函数的初始上下界计算。

Step2：计算目标函数 y 的初始上下界。其中下界 LB（Lower Bound）根据以下的公式计算：

$$LB = \max\{\frac{1}{m}\sum_{i=1}^{n} p_i; \max_i\{p_i\}; p_m + p_{m+1}\} \quad (2.39)$$

上界 UB（Upper Bound）为利用启发式规则求得的结果，根据 Mokotoff[11]给出的示例，采用 LPT 规则求得。

Step3：如果初始的 $LB = UB$，则直接跳转 step6 输出最终的结果 (x, y)，其中 x 为利用 LPT 规则求得的调度解，y 为 x 对应的最大完工时间 makespan，即为

此时上下界的值。如果不相等，则将 IPMS 问题转化为对应的松弛问题，并加入约束 $y = LB$。

Step4：根据已有的约束求解 IPMS 的松弛问题，如果松弛解为整数解，则完成计算，跳转 Step5 输出最终的结果。否则根据式(2.36)添加有效不等式，同时 $LB = LB + 1$。重复 Step4，直到无法继续产生有效不等式.。此时再根据当前的下界 LB，采用分支定界法[8]继续求解。

Step5：输出最终的结果 (x, y)。

整个割平面法的步骤的流程图如图 2.1 所示：

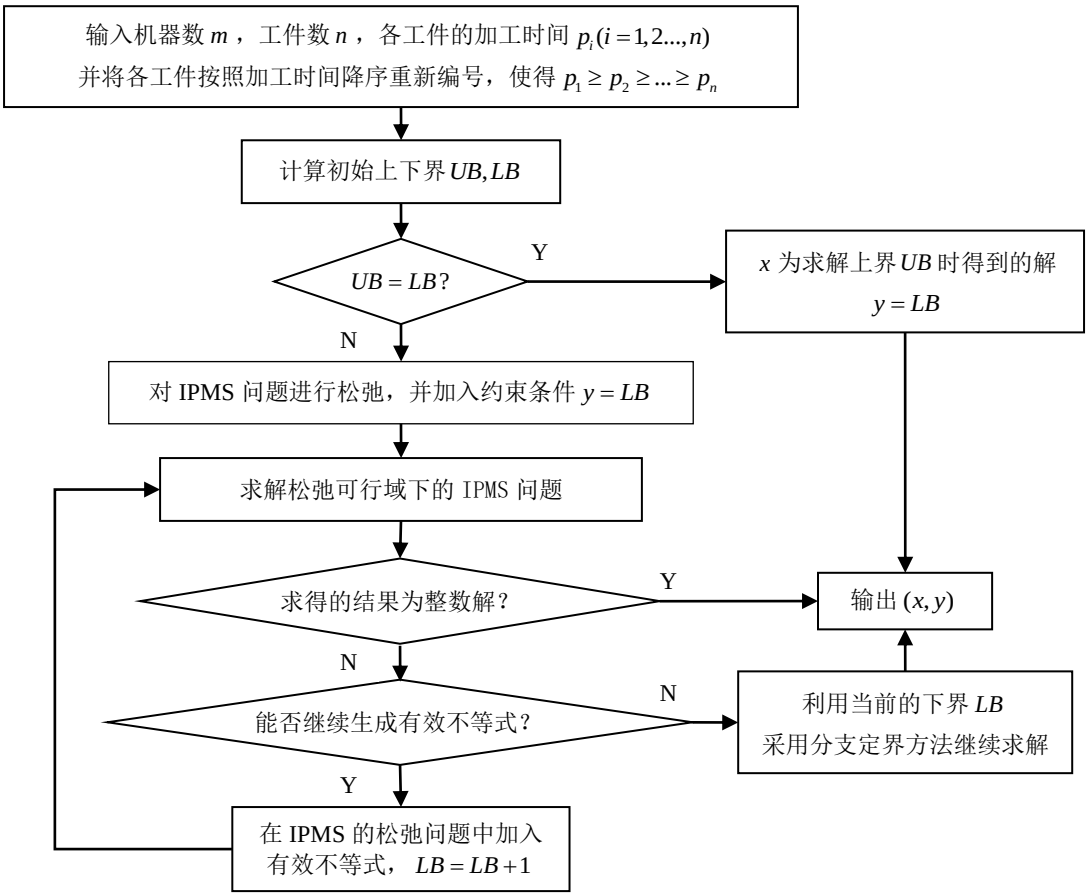


图 2.1 割平面法流程图

2.6 布谷鸟算法

布谷鸟算法（Cuckoo Search Algorithm, CSA）是一种较为新颖的群智能优化算法，由 Yang 和 Deb[55]于 2009 年提出。算法设计的灵感来源是自然界中布谷鸟

的寄生性产卵行为，并采用了莱维飞行（Lévy Flights）进行种群更新。布谷鸟算法的操作简单，全局搜索能力强，在各种优化领域包括生产调度领域都有出色的表现[56-58]。在一致并行机调度问题的求解上也有成功应用的案例[25,58]。下面将对布谷鸟算法进行简单的介绍。

2.6.1 算法简介

布谷鸟，又称杜鹃，因叫声类似“布谷”而得名，是自然界中一种常见的鸟类。同其他鸟类不同，自然界中的布谷鸟不会筑巢和育雏。它们会将蛋下在精心挑选的宿主鸟窝中，让宿主来替它们完成孵蛋和哺育的工作。布谷鸟会尽可能地对自己的蛋进行伪装，使蛋的颜色，大小，花纹等都与宿主的蛋相似，同时布谷鸟的雏鸟也会模仿宿主雏鸟的叫声来获取更多的食物。但当宿主一旦发现了布谷鸟的寄生情况，布谷鸟的蛋或者幼崽会被宿主抛弃[62]，此时布谷鸟会选择新的宿主巢穴进行寄生。

生物学家们发现，布谷鸟的飞行轨迹表现出了莱维飞行的特性.除布谷鸟外，自然界中的许多其他鸟类或者昆虫的运动行为也满足这一特征[63-67]。莱维飞行是一种方向服从均匀分布，步长满足莱维分布（Lévy Distribution）的随机游走，而莱维分布同时也是一种重尾幂律分布（Power-law Step-length Distribution with a Heavy Tail）。相比普通的随机游走，莱维飞行可以更有效地对周边环境进行搜索。

根据布谷鸟搜索宿主鸟窝进行寄生下蛋的行为和莱维飞行，Yang 和 Deb[55,68] 提出了布谷鸟算法。为了简化算法和计算，他们制定了以下三条规则：

1）每只布谷鸟每次只选择一个随机的窝下一个蛋；

2）每代中产生最优秀个体的鸟巢会被保留到下一代；

3）每代种群中会有占总数 $p_a \in [0,1]$ 伪装较差的蛋被宿主以一定概率发现抛弃，为了保持蛋的数量不变，布谷鸟会选择一个新的鸟窝重新下蛋。

根据以上规则，布谷鸟算法中每个个体对应一个鸟窝或者蛋，即为一个解，每代种群数量即为每代解的数量。鸟窝或蛋的优劣程度对应解的性能好坏。布谷鸟寻找鸟巢下蛋的过程就是在解空间里搜索新解的过程。蛋被抛弃寻找新窝的过程则为每代种群中性能较差的解以一定概率被新解代替的步骤。布谷鸟行为和利用布谷鸟算法求解优化问题可以有表 2.2 所示的映射关系：

表 2.2 布谷鸟行为与求解优化问题映射表

布谷鸟行为	求解优化问题
鸟巢/蛋	优化解
鸟巢种群大小	每代解的数量
莱维飞行寻找新巢下蛋	莱维飞行更新当前解种群
蛋被抛弃寻找新巢	选择部分坏解进行更新

布谷鸟算法中莱维飞行的步长 s 满足莱维分布，如下式所示，其中 β 为给定的常数：

$$s = Lévy(\beta) \sim u = t^{-\beta}(1 < \beta \le 3) \tag{2.40}$$

实际计算过程中采用 Mantegna 算法来产生服从莱维分布的随机步长[69]，该算法的有效性和合理性已经得到了证明[71]。产生步长的公式如下所示：

$$s = \frac{u}{|v|^{\frac{1}{\beta}}},\ 1<\beta \le 3 \tag{2.41}$$

上式中的 u, v 为两个满足正态分布的随机数，其中

$$u \sim N(0, \sigma^2),\ v \sim N(0,1) \tag{2.42}$$

随机数 u 满足的正态分布的标准差 σ 的计算公式为：

$$\sigma = \left[\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\beta\Gamma[(1+\beta)/2]\cdot 2^{(\beta-1)/2}} \right]^{1/\beta} \tag{2.43}$$

Γ 为标准的 Gamma 函数，对于给定的 β 表现为常数。π 为圆周率。

2.6.2 算法步骤

上一节介绍了布谷鸟算法的背景原理，本节将对标准布谷鸟算法的具体步骤进行介绍。

在布谷鸟算法中，用 t 表示当前的迭代代数，$P(t)$ 表示当前代种群，每代种

群包含了 PN 个解 $\mathrm{X}_i(t),\ i=1,..,PN$，$f(\mathrm{X}_i(t))$ 为 $\mathrm{X}_i(t)$ 对应的目标函数。整个布谷鸟算法的流程图如图 2.2 所示，主要包括了以下几个步骤：

Step1：随机产生 PN 个体组成初始代种群 $P(0)$ 并记录下其中性能最好的解，用 X^b 表示。

Step2：对每代种群中的每个解 $\mathrm{X}_i(t)$ 利用式(2.44)进行更新得到下一代的解 $\mathrm{X}_i(t+1)$，组成下一代种群 $P(t+1)$：

$$\mathrm{X}_i(t+1)=\mathrm{X}_i(t)+\alpha Lévy(\beta) \tag{2.44}$$

其中 $\alpha>0$ 为步长因子，通常取 $\alpha=1$[59,68]，$Lévy(\beta)$ 为满足莱维分布的随机步长 s，式(2.40)~(2.43)为其具体的计算公式。

Step3：每代种群更新完毕后，计算更新后每一个解的性能 $f(\mathrm{X}_i(t+1))$，排在后 p_a 的坏解会被随机产生的新解代替。同时根据最新种群中每个解的性能更新最优解 X^b。

更新种群和选择坏解随机产生新解代替的步骤 Step2 和 Step3 会重复执行，直到当前代数达到最大迭代次数即 $t=Maxgen$ 时停止。

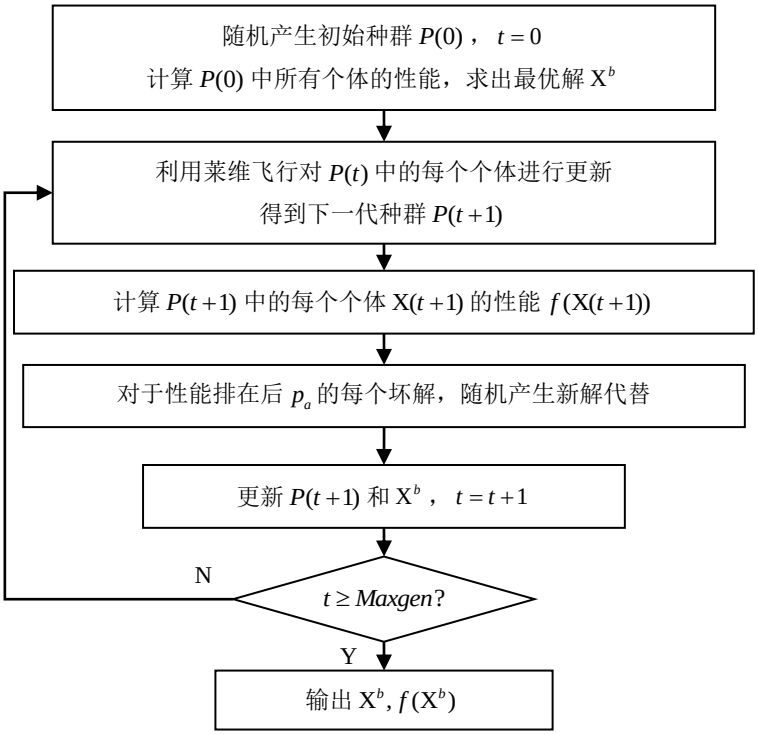


图 2.2 标准布谷鸟算法流程图

总体来说，作为一种近年来兴起的群智能优化算法，布谷鸟算法具有以下几个特点：

1）自组织性[72]。设置好算法参数和完成种群的初始化后，布谷鸟算法便不再需要外部的指令控制。整个种群中不存在领导个体，个体之间不存在互相控制，算法通过保留最优解和淘汰坏解的策略使得种群逐渐趋向最优解。

2）并行性。布谷鸟算法是一种并行搜索算法，整个种群有着共同的优化目标，不会因为个体的缺陷而受到影响。

3）全局搜索能力强。依靠莱维飞行的种群更新机制，保留最优解的机制，淘汰劣势个体机制的有机结合，布谷鸟算法具有强大的全局搜索能力。莱维飞行的飞行步长呈现频繁的短距离和偶然的长距离，相比遗传算法，粒子群算法等其他群智能优化算法，这一方式有助于算法在对解空间进行全局高效搜索的同时跳出局部最优[73]。保留最优解的方式可以对优势个体保持跟踪。而淘汰劣势个体能够在进行全局搜索时维持种群的质量。

4）易于实现，混合方便。布谷鸟算法的算法参数较少，步骤简单，公式明确。作为一种开放式算法，布谷鸟算法容易于其他搜索方式进行融合，优势互补，形成性能更好的混合算法[72]。

当然布谷鸟算法也存在着一定的不足和缺陷。布谷鸟算法主要依靠莱维飞行进行个体更新。莱维飞行的步长特征容易使个体在较大搜索范围内进行跳跃，而对于表现较坏的解只是采用了随机新解的方式进行更新，没有对当前个体的信息进行充分利用，这导致了算法的局部搜索能力较差。

2.7 一致并行机调度问题的邻域构造方式

在本文的第一章介绍过求解并行机调度问题的混合算法，通常采用将群智能优化算合和局部搜索相结合的方式来提升求解效率。通过结合群智能算法在搜索大规模解空间时的优势和局部搜索在局部深度搜索上的优点，混合算法相比单一的迭代启发式算法对解空间有更强的搜索能力。如上文介绍的布谷鸟算法，其依靠莱维飞行有着强大的全局搜索能力，但是局部搜索能力相对欠缺，加入局部搜索的步骤后可以对种群中表现较差的个体进行深度更新，利用不同搜索方式的优势互补，将布谷鸟算法引向更广的搜索空间[58-61]。

邻域构造是局部搜索中的一个关键步骤。根据所求问题或模型的特点设计面向问题（模型）的邻域构造方式，可以产生性能更佳的邻域解，从而进行更高效的局部搜索。

在一致并行机调度问题中，每个解中加工时间最长的机器被称为关键机器，其余的机器被称为非关键机器[42]，一致并行机中最常用的邻域解构造方式是通过对关键机器上的工件进行邻域操作来产生相应的邻域解。这些邻域操作可以分为交换（Swap）和插入（Insert）两种方式[24]，而交换方式又可以根据交换工件的数量分为一对一交换（Swap），非对称交换（Asymmetric Swap）和两两交换（Double Swap）。插入操作是将关键机器上的一个工件移动到另一台非关键机器上加工，一对一交换是选择关键机器上的一个工件，与另一台非关键机器上的工件交换加工位置；非对称交换将关键机器上的一个工件与某台非关键机器的两个工件进行加工位置的交换；两两交换则是交换关键机器上的两个工件与某台非关键机器上的两个工件。非关键机器以及进行插入或者交换的工件都是随机选择的。

这里以一个 7 工件 3 机器的确定性一致并行机调度问题为例展示这几种邻域操作方式，各工件的加工时间如表 2.3 所示，其中工件 J_1, J_3, J_6 在 M_1 上加工，J_2, J_4 在 M_2 上加工， J_5, J_7 在 M_3 上加工。

表 2.3 示例加工时间

工件	J_1	J_2	J_3	J_4	J_5	J_6	J_7
工件加工时间	2	4	7	5	3	3	4

图 2.3 通过甘特图展示了四种邻域解产生的过程。根据表 2.3 所示的各工件加工时间，当前解中 M_1 为关键机器， M_2 和 M_3 为非关键机器。a）图所示的插入操作将关键机器 M_1 上的工件 J_6 移动到一台非关键机器 M_3 上进行加工；b）图为一对一交换，选择 M_1 上的工件 J_3 与 M_3 上的 J_7 交换加工位置；c）图中的操作方式为非对称交换，从关键机器 M_1 上选择了一个工件 J_3 放到 M_3 上加工，再将原本

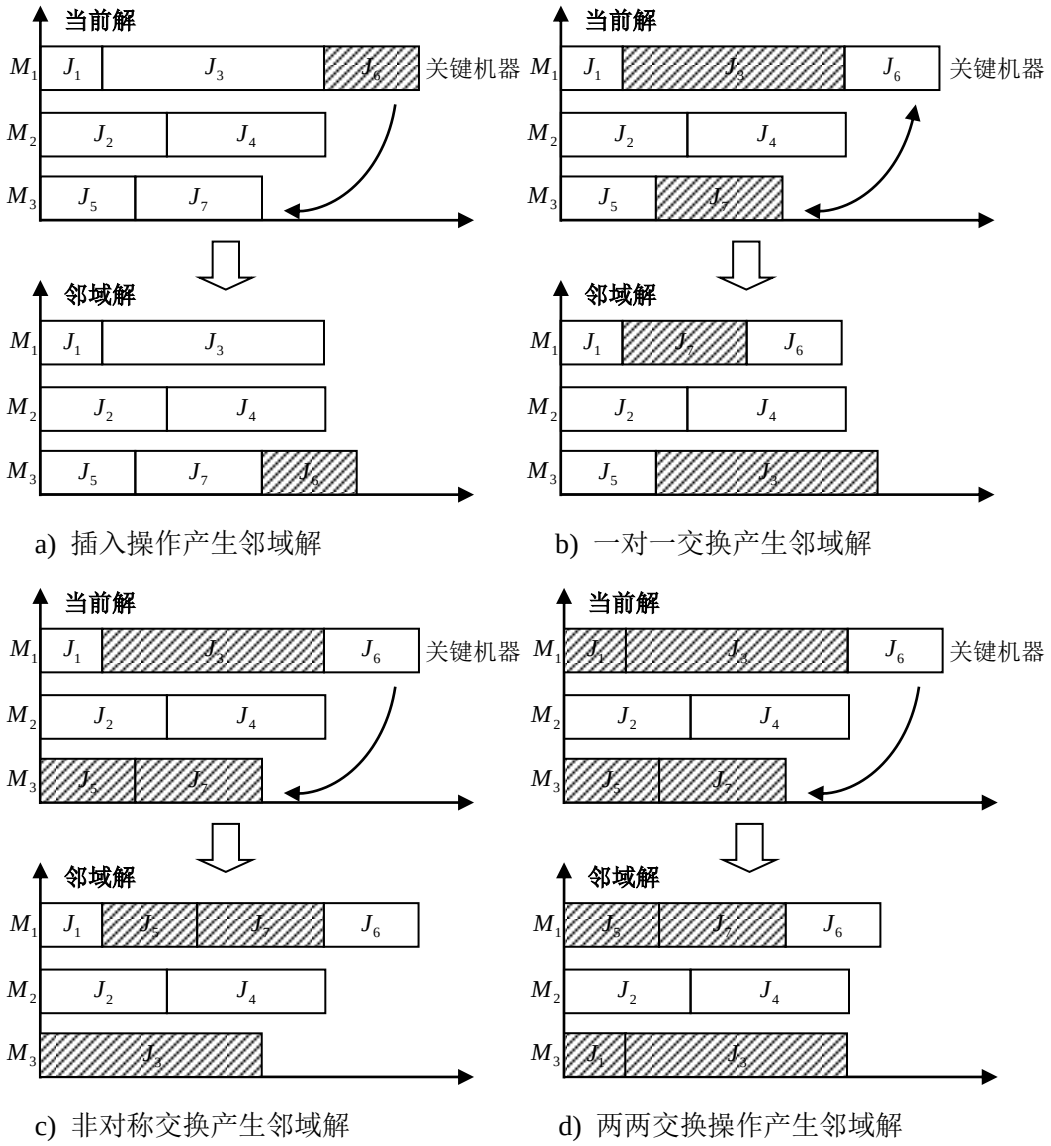


图 2.3 基于关键机器的四种邻域构造方式

在M_3上加工的两个工件J_3和J_7移到了M_1上进行加工；d）图展示了两两交换的操作过程，将M_1中的J_1, J_3与非关键机器M_3上的两个工件J_5, J_7进行位置的交换。

2.8 本章小结

本章首先介绍了一致并行机调度问题，给出了具体的定义和相关的表示符号。重点介绍了最具代表性的以最大完工时间为目标函数的确定性一致并行机调度问题，给出了其数学规划模型。然后对基于场景的不确定一致并行机调度问题进行了叙述，介绍了基于场景的随机优化模型和几种鲁棒优化模型。

除了求解模型外，本章还对求解一致并行机调度问题的精确算法和启发式算法分别进行了介绍，重点介绍了精确算法中的割平面法和启发式算法中的布谷鸟算法。对这两种算法的算法原理和基本步骤都进行了详细的介绍。然后简单提及了启发式算法中采用群智能搜索算法和局部搜索结合的混合算法，并对局部搜索中的关键步骤邻域构造方式进行了具体的描述，通过甘特图的方式展示了四种并行机调度问题中最常用的邻域构造方式。

目前对于离散场景下一致并行机调度问题的研究，在求解模型上依然以本章介绍的最坏场景模型和最大后悔场景模型为主。阈值坏场景模型可以处理鲁棒优化过程中保守性和积极性间的对立统一关系，但在一致并行机调度中尚未得到应用。在求解算法上，割平面法和布谷鸟算法在一致并行机问题的求解中都有成功的应用。割平面法作为一种精确算法，适合在求解规模较小精度要求高的场合使用。而作为一种群智能搜索算法，布谷鸟算法有着强大的全局搜索能力，对于复杂并行机问题的求解有着显著的优势，并可以通过与局部搜索混合的方式来提升其求解效率。根据以上对不确定一致并行机调度问题的介绍，本文将在下面的章节中对阈值坏场景集模型做进一步的研究讨论，尝试在一致并行机鲁棒调度问题中进行应用。同时本文将采用割平面法和混合了局部搜索的布谷鸟算法对一致并行机鲁棒调度模型进行求解，其中局部搜索的邻域构造将是在常规的一致并行机邻域基础上面向该鲁棒模型设计的全新邻域构造方式。

第三章　给定阈值下一致并行机阈值坏场景集惩罚模型的求解算法

3.1 引言

本文研究用有限个离散场景表示不确定加工时间的一致并行机调度问题（Scenario Identical Parallel Machine Scheduling, SIPMS）。本文在第二章中介绍过SIPMS 下的两阶段阈值坏场景集模型 TSPTM，该模型包含两个求解步骤，首先在第一阶段确定阈值的合理区间，然后从区间中选择一个值当作阈值，求解不同阈值下的阈值坏场景集惩罚模型PTM$|T$。通过关注更多坏场景下的性能来降低最终解的保守性。在给定不同阈值时，第二阶段的PTM$|T$可以看作是对多个阈值坏场景集惩罚模型 PTM 的求解。因此本章将对阈值给定的 PTM 模型的求解算法进行讨论研究，再在下一章对其进行延申，将本章的研究内容用于两阶段 TSPTM 中的第二阶段问题的求解。本文先根据问题特点设计了一种面向问题的合并坏场景邻域构造方式（United bad scenario Neighborhood, UN），并将使用了该邻域的局部搜索与布谷鸟算法进行混合，用该基于合并坏场景邻域局部搜索的布谷鸟算法（UN based CSA, UNCSA）对阈值坏场景集惩罚模型进行求解。

本章的 3.2 节将对本文提出的 UNCSA 算法的步骤和算法组成部分进行详细的说明，包括面向模型的邻域构造方式，布谷鸟算法与局部搜索结合的过程等。3.3 节是仿真和分析，将 UNCSA 算法和已有的其他三种智能优化算法进行对比，以此来验证算法和模型的有效性。3.4 节对本章的内容进行概括和总结。

3.2 基于合并坏场景邻域的混合布谷鸟算法 UNCSA

布谷鸟算法最初的设计多用于求解连续优化问题[69]。自 Tein 等[75]将布谷鸟算法进行改进用于解决护士排班问题后，布谷鸟算法开始在离散优化问题尤其是调度问题中得到广泛的应用[25,58]。在用于调度问题的求解时，通常需要加入离散化的步骤来实现离散调度解和连续位置解之间的转换。离散化常用的方法有最大位置规则[76]（Largest Position Value, LPV）,最小位置值规则[58]（Smallest Position

Value，SPV）等。本文将采用 SPV 规则来对布谷鸟算法进行离散化的操作，SPV 规则操作简单，已经被成功用于多种调度问题[77-79]，具体的操作方式会在下文进行详细介绍。

作为一种全局搜索能力较强的智能优化算法，布谷鸟算法中用于个体位置更新的步长服从莱维分布，这使得个体容易在较大的搜索区域内进行跳跃。虽然在每次迭代过程中会筛选出部分差解进行更新，但采用的是随机产生新解进行替换的方式，这也导致了算法的局部搜索能力较差。因而越来越多的研究人员尝试在布谷鸟算法中加入局部搜索的步骤，在保留种群中优秀个体的同时对性能较坏的个体的邻域进行深度搜索，进而提升整个种群的性能，将整个算法引向更好更广的搜索空间。混合算法通过结合两种搜索方式的特点，形成优势互补，相比单一的算法有着更高的搜索质量和效率。

本文将先对布谷鸟算法进行离散化的处理，使其能够用于一致并行机调度问题的求解，然后根据 SIPMS 下 PTM 问题的特点，设计了一种面向问题的合并坏场景邻域构造方式，并将使用了这种邻域构造方式的局部搜索与离散化后的布谷鸟算法混合用于求解。接下来将对这种混合了基于合并场景邻域局部搜索的布谷鸟算法（UN based CSA, UNCSA）进行详细的介绍和仿真测试。

3.2.1 编码和解码

在一致并行机调度问题中，主要有基于机器的编码[22]（Encode Based on Machine）和基于工作的编码[58]（Encode Based on Job）两种编码方式。基于机器的编码中的每一位数码表示具体的机器序号，因此会出现重复数码的情况。而本文采用的布谷鸟算法后续会加入离散化的步骤，需要解中每个维度的数码都不同，因此本文采用基于工件的编码方式。

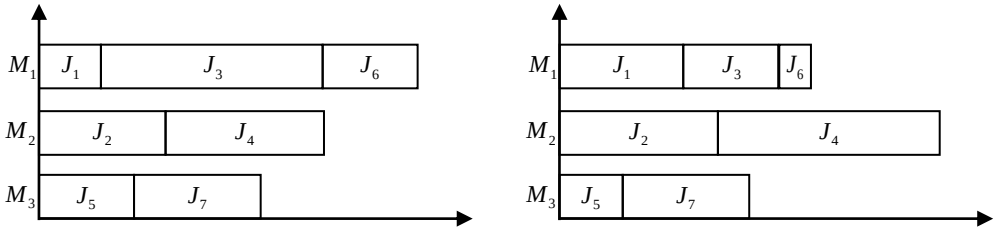
对于一个涉及 m 台机器，n 个工件问题的一致并行机问题采用基于工件的编码。机器为 $M_1 \sim M_m$，工件为 $J_1 \sim J_n$。每个解 X 由 n 位整数组成，每个数字代表对应工件的下标。在同一台机器上加工的工件按照加工顺序组成一个子集，所有子集按机器序号排列成一个解。

表 3.1 为一个 7 工件 3 机器一致并行机调度问题在两个场景下的加工时间：

表 3.1 示例加工时间

工件	J_1	J_2	J_3	J_4	J_5	J_6	J_7
λ_1下工件加工时间	2	4	7	5	3	3	4
λ_2下工件加工时间	4	5	3	7	2	1	4

如果一个调度解为$\{[1,3,6][2,4][5,7]\}$，则表示工件J_1, J_3, J_6在M_1上加工，J_2, J_4在M_2上加工，J_5, J_7在M_3上加工。其对应的在λ_1和λ_2下的生产加工甘特图如图 3.1 所示：



a) λ_1下生产加工甘特图　　b) λ_2下生产加工甘特图

图 3.1 不同场景下生产加工甘特图

3.2.2 初始种群的产生

对于群智能优化算法，良好的初始解可以提升求解的效率。LPT 规则作为一种启发式规则，可以快速产生相对优秀的一致并行机调度解，在用智能优化算法求解一致并行机调度问题时常被用作产生初始解[25]。

LPT 规则在一致并行机调度中的使用方式比较简单，即将所有工件按照加工时间非增序进行排列，然后每次将当前加工时间最长的工件安排到序号最小的空闲机器上进行加工。

以表 3.1 的加工时间为例。在场景λ_1下，七个工件按加工时间非增序进行排序如下：J_3, J_4, J_2, J_7, J_5, J_6, J_1 。一开始三台机器都处于空闲状态，先将J_3放置在M_1上进行加工，然后J_4和J_2分别在M_2, M_3上进行加工。根据各工件的加工所需时间，M_3最先进入空闲状态，下一个待加工工件J_7放置M_3上加工，J_5和J_6

分别在M_2, M_1上加工。轮到最后一个工件J_1时，M_2, M_3都处于空闲状态，优先放置在序号较小的M_2上加工。在λ_1下利用 LPT 规则得到的调度解为$\{[3,6][4,5,1][2,7]\}$，对应的甘特图如图 3.2 所示：

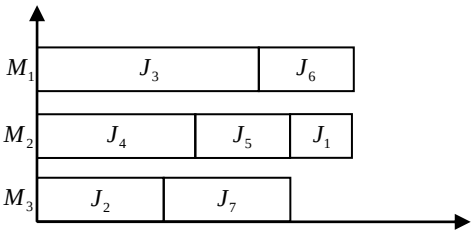


图 3.2 λ_1下 LPT 规则加工示例甘特图

在λ_2下七个工件按加工时间非增序进行排序如下：J_4, J_2, J_1, J_7, J_3, J_5, J_6 ，通过类似的操作，利用 LPT 规则得到的调度解为$\{[4,5][2,3,6][1,7]\}$，对应的甘特图如图 3.3 所示：

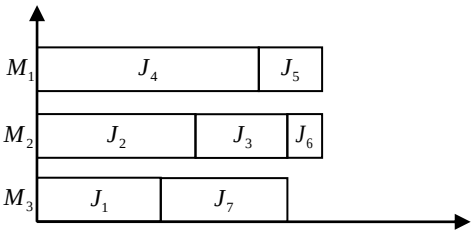


图 3.3 λ_2下 LPT 规则加工示例甘特图

本文求解$|\Lambda|$个离散场景下的一致并行机调度问题，混合布谷鸟算法 UNCSA 每代的种群大小为PN。如果算法设定的种群大小$PN \le$场景个数$|\Lambda|$时，选择在前PN个场景下利用 LPT 规则产生的解作为初始的种群$P(0)$。如果种群大小$PN >$场景个数$|\Lambda|$,则前$|\Lambda|$个初始解利用 LPT 规则生成，后$(PN - |\Lambda|)$个解则通过对前面已产生的初始解进行扰动产生。这样可以根据不同场景产生相对优秀的初始解。

3.2.3 连续化和离散化

布谷鸟算法的最初设是为了求解连续优化问题，而一致并行机调度问题的最终解为具体的工件排序，属于离散的优化问题。因此本文在标准的布谷鸟算法基

础上加入了连续化和离散化的步骤，以此来实现离散的调度解和布谷鸟算法运行过程中的连续解之间的转化。

本文中用 $P(t)$ 表示每代的调度解种群，其中的每个个体 X_i 为一个离散的调度解。用 $P'(t)$ 表示连续位置解种群，里面的每个连续解个体 X'_i 与 X_i 一一对应。在使用 LPT 规则生成初始调度解种群 $P(0)$ 后，将其连续化得到初始的连续解种群 $P'(0)$。在混合布谷鸟算法之后的迭代过程中，对每代的连续解种群 $P'(t)$ 中的个体采用莱维飞行更新得到下一代的 $P'(t+1)$，再通过对 $P'(t+1)$ 进行离散化得到下一代的调度解种群 $P(t+1)$。

对于初始调度解种群 $P(0)$ 中的个体 $\mathrm{X}=(x_1, x_2, ..., x_n)$，其对应的连续解 $\mathrm{X}'=(x'_1, x'_2, ..., x'_n)$ 通过下式计算得到：

$$x'_i = \frac{1}{n} \cdot x_i, \quad i = 1, 2, ..., n \tag{3.1}$$

本文将连续化的结果保留 3 位有效数字。以一个调度解 $\{[1,7][2,4,6][5,3]\}$ 为例，按式(3.1)进行计算，则 $x'_1 = \frac{1}{7} \cdot 1 = 0.142$，$x'_2 = \frac{1}{7} \cdot 7 = 1$，..., $x'_7 = \frac{1}{7} \cdot 3 = 0.428$。最终对应的连续解为 $\{[0.142,\ 1.00][0.285,\ 0.571,\ 0.857][0.714,\ 0.428]\}$。

每一代中的连续解进行离散化转为调度解的步骤根据 SPV 规则实现。SPV 规则的操作步骤比较简单，对于一个连续解 X'，先将 X' 的各位数值 $x'_1 \sim x'_n$ 按照升序进行排序，排序后位于第 i 位的数值在原本 X' 中的位置序号即为对应的离散解 X 的 x_i。表 3.2 为 SPV 规则应用的一个具体例子，假设某个连续解 $\mathrm{X}'=\{[0.16,\ 2.89,\ 3.56][2.77, 0.64][0.59,\ 1.32]\}$，先对各位数值按升序进行排序，得到 b）表所示的结果。排序后的第一位 0.16，为 X' 中 x'_1 的数值，则对应的离散调度解的第一位 $x_1 = 1$，排序后的第二位为原本 X' 中 x'_6 的数值 0.59，则 $x_2 = 6$。以此类推，则该连续解 X' 利用 SPV 规则离散化后得到的离散调度解 $\mathrm{X}=\{[1,\ 6,\ 5][7,\ 4][2,\ 3]\}$。

表 3.2　SPV 规则举例

a) 连续解 X′：

序号 i	1	2	3	4	5	6	7
x_i'	0.16	2.89	3.56	2.77	0.64	0.59	1.32

b) X′ 中的各位数值按升序排列：

排序顺序	1	2	3	4	5	6	7
x_i'	0.16	0.59	0.64	1.32	2.77	2.89	3.56
原始序号	1	6	5	7	4	2	3

c) 转换后调度解 X：

序号 i	1	2	3	4	5	6	7
x_i	1	6	5	7	4	2	3

3.2.4　面向阈值坏场景集惩罚模型的合并坏场景邻域

本章将采用在布谷鸟算法中加入局部搜索的方式对 SIPMS 的 PTM 问题进行求解。邻域构造作为局部搜索中的关键步骤，影响着局部搜索的效率，因此先对邻域构造方式进行讨论研究。PTM 的优化目标是所有坏场景下性能与阈值差值的惩罚值，求解过程涉及多个坏场景。所以先对单一场景下的邻域构造进行研究。而单一场景下的 SIPMS 问题可以看作是确定性一致并行机问题。因此本文先参考已有的确定性一致并行机调度问题设计单一坏场景下的邻域构造方式，再根据 SIPMS 下的 PTM 问题的特点，设计一种面向问题的合并场景邻域用于构造局部搜索中的邻域解。

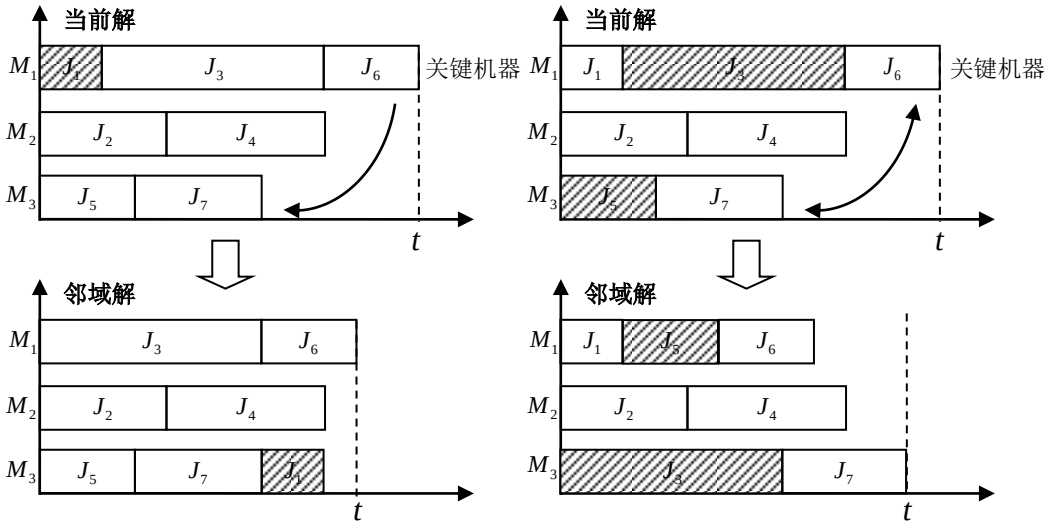
在第二章的 2.7 节中，本文介绍了一致并行机调度问题中常用的邻域构造方式。这几种方式通常随机选择进行交换或者插入的工件。相比于其他的并行机调度问题，一致并行机调度中同一个工件在不同的机器上的加工时间是保持不变的。根据这个特点，本文在原有的邻域构造方式基础上进行拓展，设计了两种面向一致并行机调度问题的邻域构造方式，更加有针对性的选择进行插入和交换的工件。

第一种邻域构造基于原本的插入方式。在进行插入操作时，选择关键机器上加工时间最短的 J_i，插入到当前总加工时间最短的非关键机器的加工队列中进行加工。第二种邻域构造采用一对一交换的方式，选择关键机器上加工时间最长

的 J_k，与总加工时间最短的非关键机器上加工时间最短的 J_j 进行加工位置的交换。

以图 3.1 中的调度解 $\{[1,3,6][2,4][5,7]\}$ 作为当前解，采用本文提出的方式选择工件进行插入或交换产生邻域解。以表 3.1 中 λ_1 下的各工件加工时间为例，当前解的关键机器为 M_1，加工时间最短的非关键机器为 M_3。则选择 M_1 上加工时间最短的 J_1，插入到 M_3 上进行加工。而在进行交换操作时，选择关键机器 M_1 上加工时间最长的 J_3，与 M_3 上加工时间最短的 J_5 进行交换得到邻域解。

图 3.4 展示了利用本文设计的方式选择工件产生邻域解的过程。从图中可以看出邻域解相比当前解在最大完工时间上的改善。相比随机选择工件，本文提出的操作方式能让加工时间的分配更加均匀，从而减少当前解的最大完工时间，产生相对优秀的邻域解。



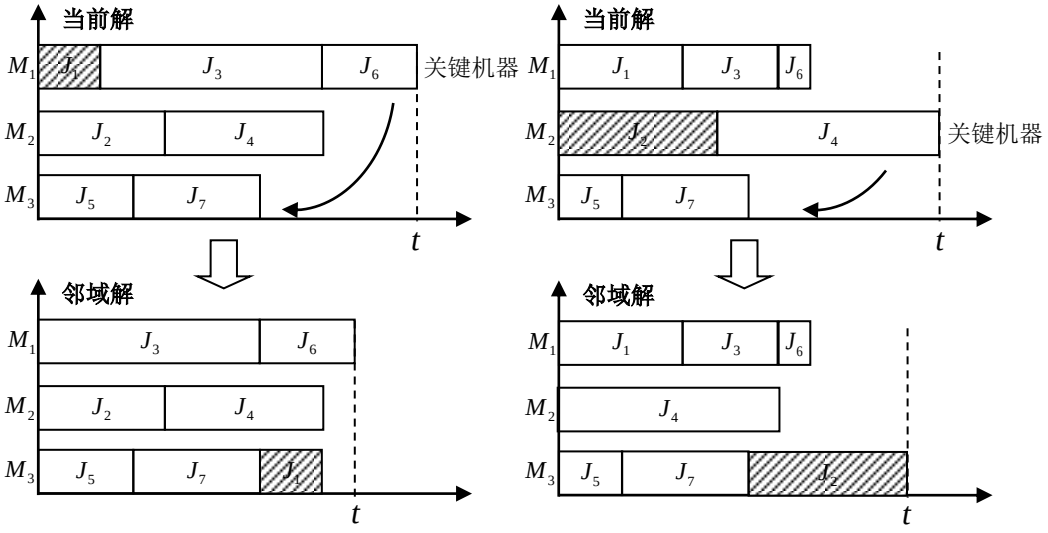
a) 插入操作产生邻域解　　b) 一对一交换产生邻域解

图 3.4 单场景下面向一致并行机问题的邻域构造方式

本文求解 SIPMS 下的 PTM 问题，需要考虑同一个解在不同坏场景下的表现性能。但由于不同场景下加工时间不同，同一个解在不同场景下呈现出不同的关键机器，导致即使采用相同规则的操作方式也会产生完全不同的邻域解。

以表 3.1 中两个场景下的加工时间为例，当前加工序列与图 3.4 中的当前解相同。图 3.5 展示了不同场景下进行插入操作产生邻域解的过程。根据表 3.1 的加工时间，λ_1 下的关键机器为 M_1，对应的加工时间最短的工件为 J_1，插入操作

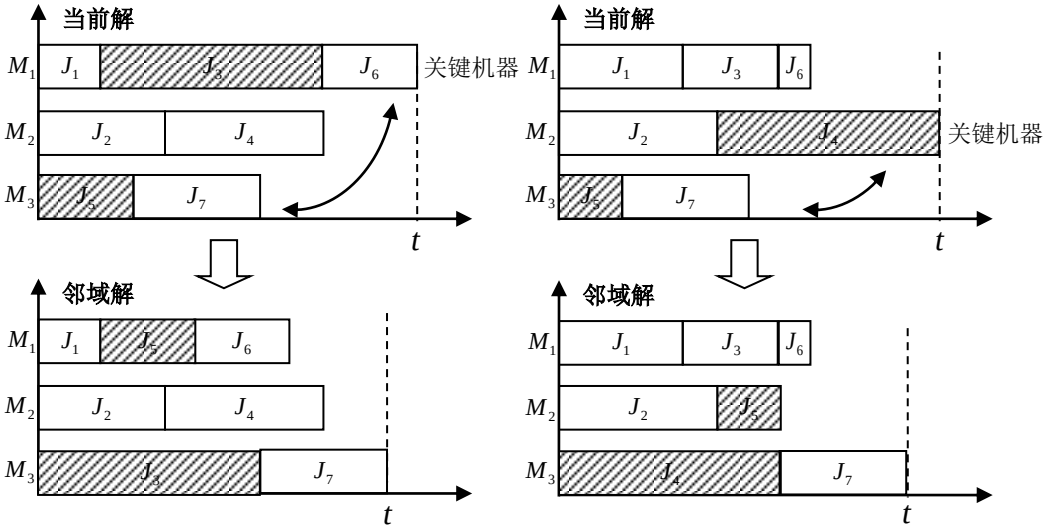
是将 J_1 移动到总加工时间最短的 M_3。λ_2 下的关键机器为 M_2，则选择将 M_2 上的 J_2 插入到总加工时间最短的 M_3 上加工。



a) λ_1 下插入操作产生邻域解　　b) λ_2 下插入操作产生邻域解

图 3.5 不同场景下插入操作产生邻域解

图 3.6 展示了不同场景下的交换操作。在 λ_1 下，选择关键机器 M_1 上加工时间最长的 J_3，与 M_3 上加工时间最短的 J_5 交换加工位置。而在 λ_2 下的关键机器为 M_2，总加工时间最短的非关键机器为 M_3。则将 M_2 上加工时间最长的工件 J_4 与



a) λ_1 下交换操作产生邻域解　　b) λ_2 下交换操作产生邻域解

图 3.6 不同场景下交换操作产生邻域解

M_3上加工时间最短的J_5交换加工位置。通过图 3.5 和图 3.6，可以清楚的看到即使采用同一种邻域操作规则，同一个当前解在不同的场景下也会产生完全不同的邻域解。因此在求解 PTM 模型时需要充分考虑不同场景对邻域操作造成的影响。

Wang 等[45]在求解离散场景下的作业车间调度问题时，在单一场景下基于关键路径这一概念来构造邻域。与一致并行机基于关键机器构造邻域解相似，在不同的场景下同一个调度解会呈现出不同的关键路径，导致产生完全不同的邻域解。为了克服不同场景对邻域解造成的影响，Wang 等[45]提出了合并场景邻域这一概念，将所有单一场景下的邻域解集合起来形成合并场景邻域。

定义 3.1：对于一个解 X，其在场景λ下产生的邻域解用$N(\lambda,\mathrm{X})$表示，合并场景邻域$UN(\Lambda,\mathrm{X})$表示 X 在场景集Λ下的所有场景产生的邻域集合：

$$UN(\Lambda,\mathrm{X})=\cup_{\lambda\in\Lambda}N(\lambda,\mathrm{X}) \tag{3.2}$$

其中不同场景下产生的重复邻域解只会保留一个，多余的相同解会被删除。

参照这一做法，本文在上文提出的两种单场景一致并行机邻域解构造方式的基础上，根据 PTM 模型的特点，围绕坏场景集$\Lambda_T(\mathrm{X})$建立面向 SIPMS 下 PTM 问题的合并坏场景邻域。

本文中用$IO(\lambda,\mathrm{X}),SO(\lambda,\mathrm{X})$分别表示 X 在场景$\lambda$下采取插入（Insert Operation）和交换操作（Swap Operation）后产生的邻域解，则$N(\lambda,\mathrm{X})=\{IO(\lambda,\mathrm{X}),SO(\lambda,\mathrm{X})\}$。X 在其坏场景集$\Lambda_T(\mathrm{X})$下的合并坏场景邻域用$UN(\Lambda_T(\mathrm{X}))$表示：

$$UN(\Lambda_T(\mathrm{X}))=\cup_{\lambda\in\Lambda_T(\mathrm{X})}N(\lambda,\mathrm{X}) \tag{3.3}$$

同样在$UN(\Lambda_T(\mathrm{X}))$中，不同坏场景下产生的重复候选邻域解会被删除。

整个基于坏场景集产生合并坏场景邻域的伪代码如图 3.7 所示。

相比单一场景下的邻域构造，合并坏场景邻域将不同坏场景下的邻域综合起来进行考虑。这一操作方式扩大了每代邻域解的个数，同时充分考虑了不同场景对邻域操作造成的影响，使算法在局部搜索的过程中有更大的几率获得性能更好的邻域解。

Procedure 1: 构建坏场景集下的合并坏场景邻域

Input: 调度解 X 和其坏场景集 $\Lambda_T(\mathrm{X})$
Output: X 的合并坏场景邻域 $UN(\Lambda_T(\mathrm{X}))$
Begin:
Step1：对每一个场景 $\lambda \in \Lambda_T(\mathrm{X})$
确定 X 在 λ 下的关键机器，通过插入和交换操作得到 $N(\lambda,\mathrm{X})$
Step2：初始化 $UN(\Lambda_T(\mathrm{X}))=\varnothing$
Step3：集合 X 在所有坏场景集下的邻域解，形成合并场景邻域
$UN(\Lambda_T(\mathrm{X}))=\cup_{\lambda\in\Lambda_T(\mathrm{X})}N(\lambda,\mathrm{X})$
Step4：对于合并场景邻域中的每个解 $\mathrm{X}_i \in UN(\Lambda_T(\mathrm{X}))$
对于合并场景邻域中的其他解 $\mathrm{X}_j \in UN(\Lambda_T(\mathrm{X})), j\neq i$
如果 $\mathrm{X}_j=\mathrm{X}_i$，删除 X_j
Step5：输出 $UN(\Lambda_T(\mathrm{X}))$
End

图 3.7 合并坏场景邻域伪代码

3.2.5 基于合并坏场景邻域的局部搜索

在上一小节中对本文设计的面向 SIPMS 下的 PTM 问题的合并坏场景邻域构造进行了介绍，本小节中介绍混合布谷鸟算法 UNCSA 中对该邻域进行局部搜索的具体步骤。

在 UNCSA 中，每一次迭代过程中会筛选出性能表现较差的后 p_a 的解。用 X^{bad} 表示一个坏解，首先确定 X^{bad} 的坏场景集 $\Lambda_T(\mathrm{X}^{bad})$，然后根据 3.2.4 中的合并坏场景邻域产生步骤形成 X^{bad} 在其所有坏场景下的合并坏场景邻域 $UN(\Lambda_T(\mathrm{X}^{bad}))$。接着将 $UN(\Lambda_T(\mathrm{X}^{bad}))$ 中的所有候选邻域解按性能排序，选出其中表现性能最好的邻域解，记作 X_{best}^{bad}。给定一个概率 p_0，如果邻域解 X_{best}^{bad} 的性能优于进行局部搜索的当前解 X^{bad}，则用 X_{best}^{bad} 代替 X^{bad}。否则以 p_0 的概率将 X^{bad} 更新为 X_{best}^{bad}。

这一利用基于合并坏场景邻域的局部搜索步骤记作 $LS(UN(\Lambda_T(\mathrm{X}^{bad})))$（Local Search based UN），整个局部搜索的具体操作如图 3.8 的伪代码所示：

Procedure：合并场景邻域下的局部搜索 $LS(UN(\Lambda_T(\mathrm{X}^{bad})))$

Input: 当前解 X^{bad} 和其坏场景集下的合并场景邻域 $UN(\Lambda_T(\mathrm{X}^{bad}))$
Output: 局部搜索后的邻域解 X^{bad}
Begin:
Step1：计算 $UN(\Lambda_T(\mathrm{X}^{bad}))$ 中每个解的性能，选出合并场景邻域中性能最好的解 X_{best}^{bad}
Step2：如果 $f(\mathrm{X}_{best}^{bad}) > f(\mathrm{X}^{bad})$
$\mathrm{X}^{bad} = \mathrm{X}_{best}^{bad}$
Step3：否则产生一个随机数 r
如果 $r < p_0$
$\mathrm{X}^{bad} = \mathrm{X}_{best}^{bad}$
Step4：输出邻域解 X^{bad}
End

图 3.8 $LS(UN(\Lambda_T(\mathrm{X}^{bad})))$ 伪代码

3.2.6 终止准则

在智能优化算法中，终止准则起着平衡求解效率和求解结果的作用。合适的终止准则可以使算法在合理的时间成本消耗下获得性能相对优秀的解。本文采用的 UNCSA 算法的主框架是布谷鸟算法，参考其他已有的布谷鸟算法[25]，采用最大迭代次数 *Maxgen* 作为算法终止条件。但 *Maxgen* 具体的数值需要结合实际问题的计算效果才能决定，本文会在下文的仿真实验中确定 *Maxgen* 的具体数值。

3.2.7 算法步骤

综合上面几小节的内容，整个用于求解 SIPMS 下的 PTM 问题的 UNCSA 算法总流程图如图 3.9 所示，包含了以下几个步骤：

Step1：输入阈值 T，场景集 Λ，初始化迭代次数 t，最大迭代次数 *Maxgen* 等算法参数。

Step2：利用 3.4.2 节的方法产生初始调度解种群 $P(0)$，并记录下种群中的最优解 X^b。将 $P(0)$ 中的每个个体进行连续化得到初始的连续解种群 $P'(0)$。

Step3：利用莱维飞行对当前代的连续解种群进行更新产生下一代连续种群 $P'(t+1)$。同时同 $P'(t+1)$ 使用 SPV 规则得到下一代调度解种群 $P(t+1)$。

Step4：计算 $P(t+1)$ 中各解的性能，筛选出性能较差的后 p_a 个坏解。计算每个坏解的坏场景集，构造其合并坏场景邻域。

Step5：对每个坏解 X^{bad} 的合并坏场景邻域 $UN(\Lambda_T(X^{bad}))$ 执行 $LS(UN(\Lambda_T(X^{bad})))$ 进行局部搜索，更新 $P(t+1)$。并将对 $P(t+1)$ 中调度解个体所作的交换插入操作同步到 $P'(t+1)$ 的对应个体中，对 $P'(t+1)$ 也进行更新。并根据更新后的种群判断是否要更新最优解 X^b。同时迭代次数 $t=t+1$。

Step6：重复 Step3~Step5，直到满足终止准则 $t=Maxgen$ 后停止迭代输出最终结果。

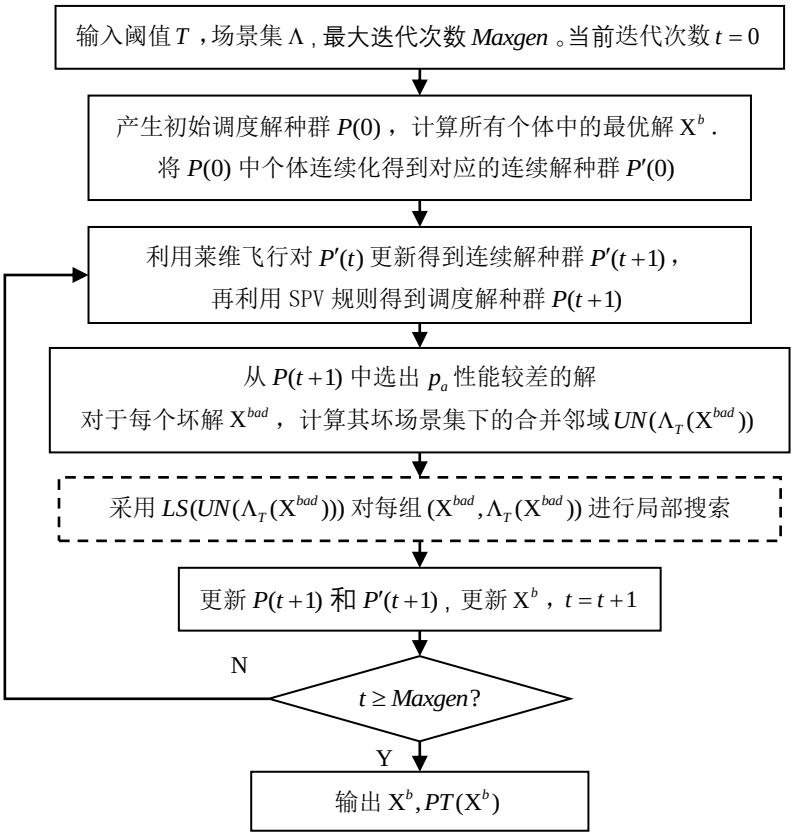


图 3.9 UNCSA 算法流程图

3.3 仿真实验与分析

本节中将采用 UNCSA 算法来求解离散场景下加工时间不确定的一致并行机调度问题的阈值坏场景集惩罚模型，优化目标为所有坏场景下的惩罚值之和。首先对算法和算例的相关参数进行测试设置，然后将 UNCSA 算法与采用了普通局部搜索的布谷鸟算法 ICSA[25]，采用了遗传算法加局部搜索的 GA+LS[47]，还有传统的模拟退火算法 SA[42]进行比较，通过大量的仿真实验数据来展示在求解 SIPMS 下的 PTM 问题时布谷鸟算法和本文设计合并坏场景邻域的有效性和优

势，以此表明本文提出的 UNCSA 算法的合理性和高效性。

本文中的所有仿真程序均采用 C++语言编写实现，在 Windows10 操作系统环境下采用 Visual Studio 2017 软件运行。仿真所用计算机的 CPU 为 Intel（R） Xeon（R） Gold 5188 ,运行内存为 256G。

3.3.1 算例设置

参照文[11,25]中的测试算例设置，本文设置了 30 种不同规模（$m*n$）的算例，并将其分为三种类型：小规模算例 E_1 $(m*n\le 100)$，中规模算例 E_2 $(100<m*n\le 1000)$ 和大规模算例 E_3 $(m*n>1000)$，每种类型包含 10 个不同规模的算例。同时本文还将参考 Kouvelis 等[80]产生离散场景的方式来对加工时间进行不确定化处理。每个场景 λ 下工件 J_i 的加工时间 $p_i^{\lambda}, i=1,..,n,\ \lambda\in\Lambda$ 从区间 $[10,10+100t_\beta]$ 中产生，其中 t_β 为控制加工时间取值的参数，在 (0.4, 0.6, 0.8, 1.0) 中选取。测试算例的规模和加工时间的设置如表 3.3 所示：

每个算例的加工时间涉及 16 个场景，即 $|\Lambda|=16$。每组算例组合 (m,n,t_β) 进行 20 次测试，一共进行 2400 个算例测试。

表 3.3 算例参数设置

	m	n	p_i^{λ}
E_1	3	5, 8, 10, 15, 20	$p_i^{\lambda}\in[10,10+100\beta]$, $\beta\in\{0.4,0.6,0.8,1.0\}$
	5	10, 12, 15, 18, 20	
E_2	3, 5	50, 100, 200	
	10	25, 50, 100	
	15	20	
E_3	3, 5	500	
	10	200, 500	
	15	100, 200, 500	
	100	200, 500, 1000	

3.3.2 算法参数设置

整个 UNCSA 算法需要设置以下几个算法参数，每代的种群大小 PN，莱维飞行时的步长因子 α，计算莱维飞行时所用的常数 β，每次迭代后筛选出的坏解百分比 p_a，局部搜索时接受性能较差的邻域解的概率 p_0 以及最大迭代次数

Maxgen。

考虑到初始解的产生方式，本文中的种群大小设置为和场景数相同，即 $PN=|\Lambda|=16$，参照文[55,59]中对布谷鸟算法参数的设置，步长因子 $\alpha=1$，$\beta=1.5$。每代选择 $p_a=0.25$ 的坏解进行局部搜索。局部搜索时性能较差的邻域解接受概率 $p_0=0.5$。而对于最大迭代次数 *Maxgen* 的设定，在下文中会通过具体的仿真确定。

3.3.3 UNCSA 算法终止准则对求解结果的影响

根据上一节的算法参数设置，本节中将对 UNCSA 的终止准则开展研究。从 $E_1\sim E_3$ 三种规模类型中各选取两个规模的算例，包括 E_1 中的 $(m*n=3*20,\ 5*20)$，E_2 中的 $(m*n=5*50,\ 10*50)$，E_3 中的 $(m*n=15*100,\ 100*200)$，分别记录下使用 UNCSA 在不同迭代次数下对 PTM 模型的求解结果。对于每组参数组合 (m,n,t_β) 的 20 次仿真测试，选择目标函数 *PT* 性能值最优的那次进行展示。测试的结果如表 3.4 所示，其中的 T 为不同参数组合 (m,n,t_β) 对应的阈值 T，*PT* 为使用 UNCSA 求解 PTM 模型得到的最优解 X^b 对应的性能值 $PT(\mathrm{X}^b)$，*CPU* 为整个求解过程消耗的 CPU 时间。

从表 3.4 中可以看出，在不同的问题规模下，最大迭代次数 *Maxgen* 的增加会增加 UNCSA 算法求解过程中的消耗时间，但同时也会提升 UNCSA 算法的最终求解结果的性能。在迭代次数相对较少时，UNCSA 的求解结果随着代数的增加改善比较明显。而在算法运行到一定的迭代次数后解的改善开始变得缓慢，此时再继续运行 UNCSA，只会增加大量的时间消耗而不能显著地提升求解效果。为了更清晰地观察这种趋势，本文从 E_1~E_3 三种不同规模类型中各选取两个规模的算例，包括 E_1 中的 $(m*n=3*20,\ 5*20)$，E_2 中的 $(m*n=5*50,\ 10*50)$，E_3 中的 $(m*n=15*100,\ 100*200)$，，绘制了其在时间取值参数 $t_\beta=0.8$ 时 CPU 消耗时间随迭代次数 *Generation* 的变化曲线以及 *PT* 值和迭代次数 *Generation* 的收敛图，如图 3.10~图 3.15 所示。通过观察 CPU 消耗随迭代次数变化的曲线可以得出，在不同规模的算例中，随着迭代次数的增加，UNCSA 算法完成求解需要的时间基本呈线性增加。而结合表 3.4 和不同算例的求解过程收敛图可以发现，最终的

求解结果大约在 250 代左右开始进入收敛。此时继续运行算法会牺牲大量的 CPU 时间但并不能对最终的求解结果进行明显的改善。将 UNCSA 的最大迭代次数 *Maxgen* 设置为 250 代以上并没有太大意义。综合考虑求解质量和 CPU 时间的消耗，本文中的 UNCSA 算法的终止准则将设置为最大迭代次数 $Maxgen = 250$ 。

同时，通过观察表 3. 4 还可以发现，参数 t_β 对算法的影响主要表现为通过改变加工时间的取值范围来改变最终解的目标函数值。而对于算法本身而言，虽然 t_β 的增大会使得工件的加工时间取值分布空间范围增大，增加整个系统的不确定性，从而略微增加 CPU 时间的消耗，对于实际的算法运行趋势和规律没有影响。

表 3.4 不同迭代次数下 UNCSA 求解结果对比

	m	*n*	β	*T*	*Maxgen* = 150		200		250		300		350		400	
					CPU	*PT*	*CPU*	*PT*	*CPU*	*PT*	*CPU*	*PT*	*CPU*	*PT*	*CPU*	*PT*
E_1	3	20	0.4	**193.31**	15.77	47.74	20.36	42.33	25.33	38.33	30.41	38.23	35.09	38.23	40.18	38.23
			0.6	**267.69**	15.83	49.19	20.52	44.00	26.73	40.68	30.51	39.94	35.39	38.77	41.57	38.52
			0.8	**340.25**	16.02	56.78	21.55	50.85	27.37	48.70	31.03	46.20	36.86	45.37	41.91	45.37
			1	**422.25**	16.81	87.19	21.87	65.63	27.60	59.74	31.60	57.14	37.41	56.48	42.03	55.97
	5	20	0.4	**107.81**	20.64	61.48	24.74	55.17	29.66	49.89	35.40	48.07	39.81	48.07	44.44	48.07
			0.6	**145.38**	20.85	90.01	24.79	76.41	29.82	66.10	35.63	64.23	40.65	63.25	45.70	62.89
			0.8	**188.75**	20.91	111.63	25.68	79.36	30.36	68.89	36.03	65.05	40.75	64.15	46.33	64.00
			1	**246.25**	21.05	120.15	25.89	104.89	30.78	75.30	36.25	73.64	40.80	72.94	46.38	72.85
E_2	5	50	0.4	**295.75**	29.17	84.40	37.45	79.97	47.24	71.07	58.86	69.61	67.04	68.15	76.79	68.15
			0.6	**387.56**	30.56	142.59	37.69	101.12	47.35	95.22	59.00	94.80	68.38	94.60	77.76	93.89
			0.8	**515.56**	30.62	141.04	38.79	111.53	48.60	100.48	59.66	99.29	69.59	98.78	78.43	98.10
			1	**623.50**	30.79	158.90	39.72	127.31	49.16	115.93	60.56	110.31	70.03	109.54	80.51	109.15
	10	50	0.4	**139.13**	39.66	125.43	48.71	106.06	57.82	80.75	68.59	79.75	77.27	79.25	87.56	79.25
			0.6	**202.94**	39.99	113.00	49.17	97.58	58.23	84.41	68.84	83.89	77.38	82.79	88.91	82.55
			0.8	**256.50**	40.92	167.63	50.65	143.80	58.98	131.82	69.64	128.46	79.13	127.76	89.97	126.54
			1	**309.13**	41.72	184.94	51.20	160.51	60.53	138.37	72.20	137.18	81.61	136.80	91.11	136.10
E_3	15	100	0.4	**191.94**	63.20	176.17	83.43	157.55	105.33	122.01	125.09	120.98	146.25	119.35	166.22	118.11
			0.6	**269.50**	63.93	186.53	84.36	174.65	105.80	161.54	126.09	158.47	147.40	157.43	167.34	157.43
			0.8	**341.81**	64.44	206.31	86.69	181.22	106.66	162.07	126.55	160.27	147.74	158.98	168.67	157.21
			1	**407.44**	64.82	354.12	86.93	272.40	107.58	200.59	127.02	199.65	148.85	199.65	169.76	198.35
	100	200	0.4	**42.19**	116.11	47.98	151.25	40.68	188.35	36.69	223.73	35.31	263.35	35.31	299.07	35.31
			0.6	**63.25**	118.21	71.41	153.09	64.51	191.57	59.90	223.88	59.78	264.08	59.78	300.77	59.67
			0.8	**84.31**	118.71	97.36	155.15	84.54	190.61	82.22	225.82	81.43	267.14	80.42	301.39	80.42
			1	**105.63**	120.61	129.31	157.30	118.47	192.06	108.27	227.35	106.74	269.83	105.56	302.47	104.32

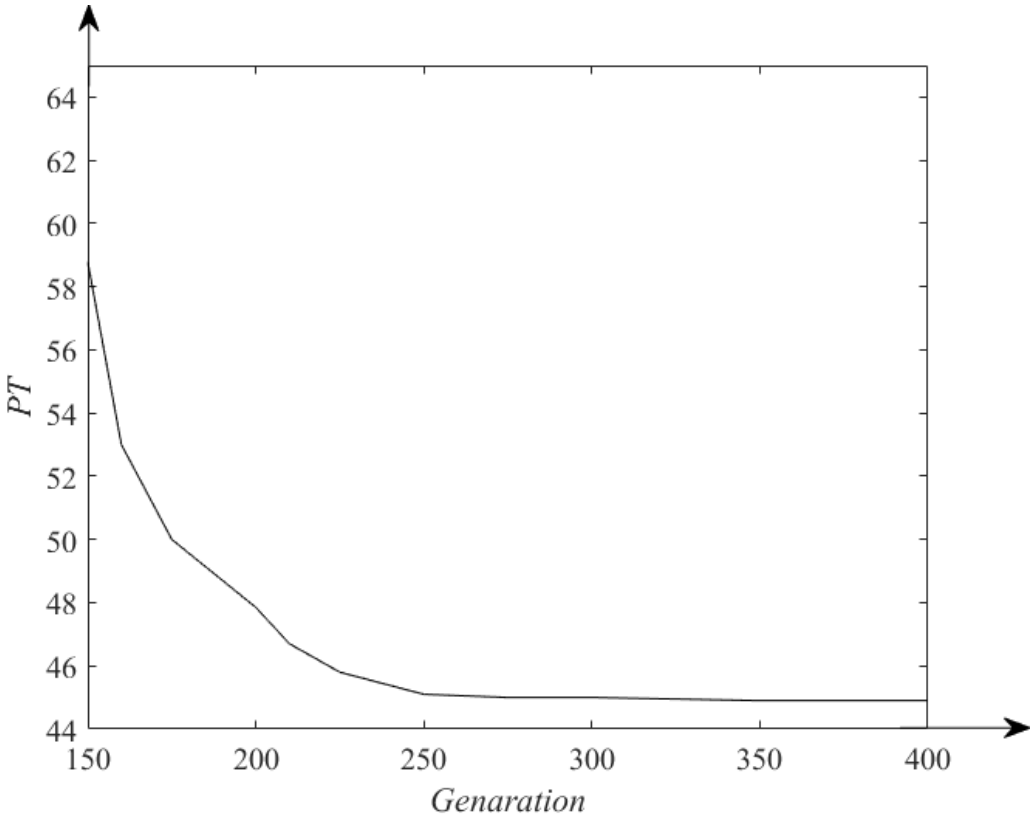


图 3.10 UNCSA (m=3,n=20) 算例迭代次数与 PT 的收敛图

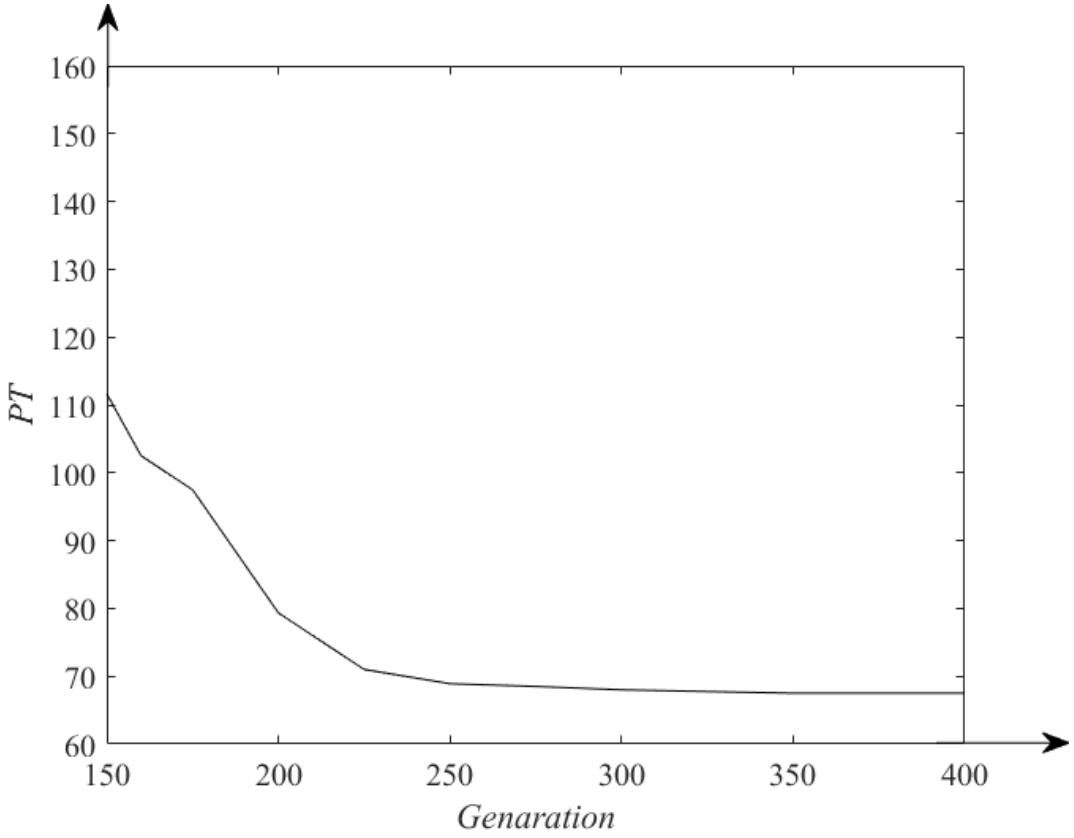


图 3.11 UNCSA (m=5,n=20) 算例迭代次数与 PT 的收敛图

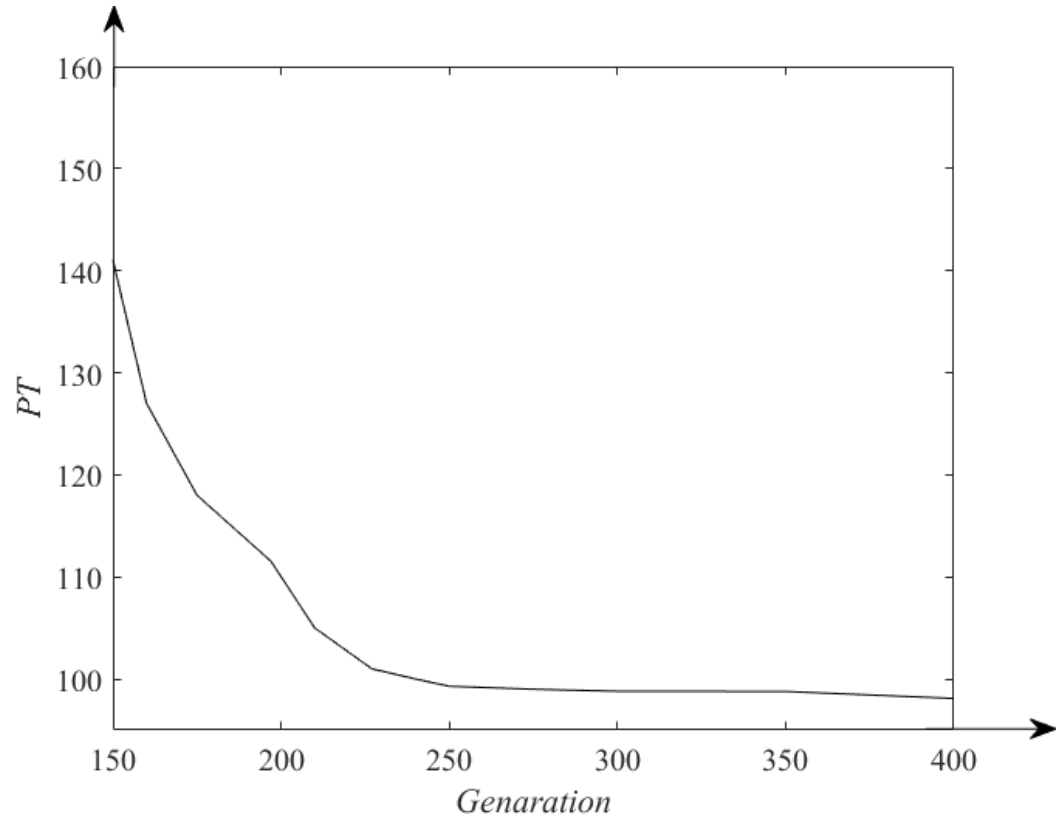


图 3.12 UNCSA (m=5,n=50) 算例迭代次数与 PT 的收敛图

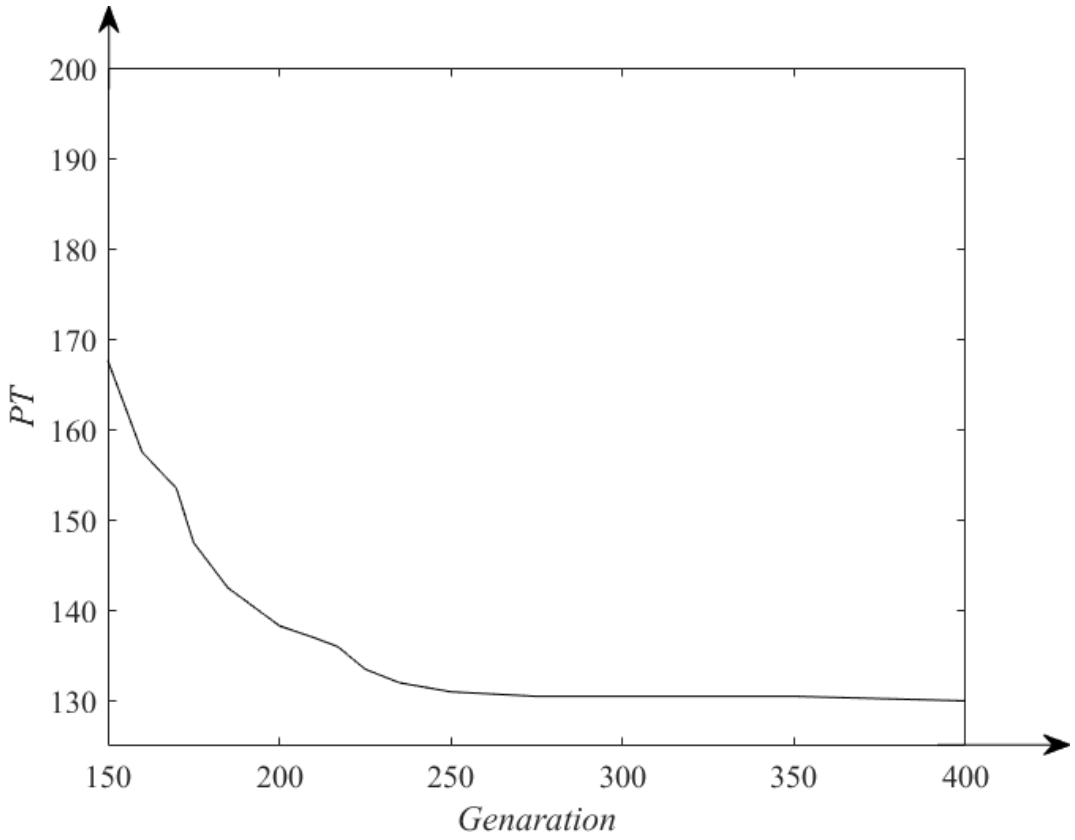


图 3.13 UNCSA (m=10,n=50) 算例迭代次数与 PT 的收敛图

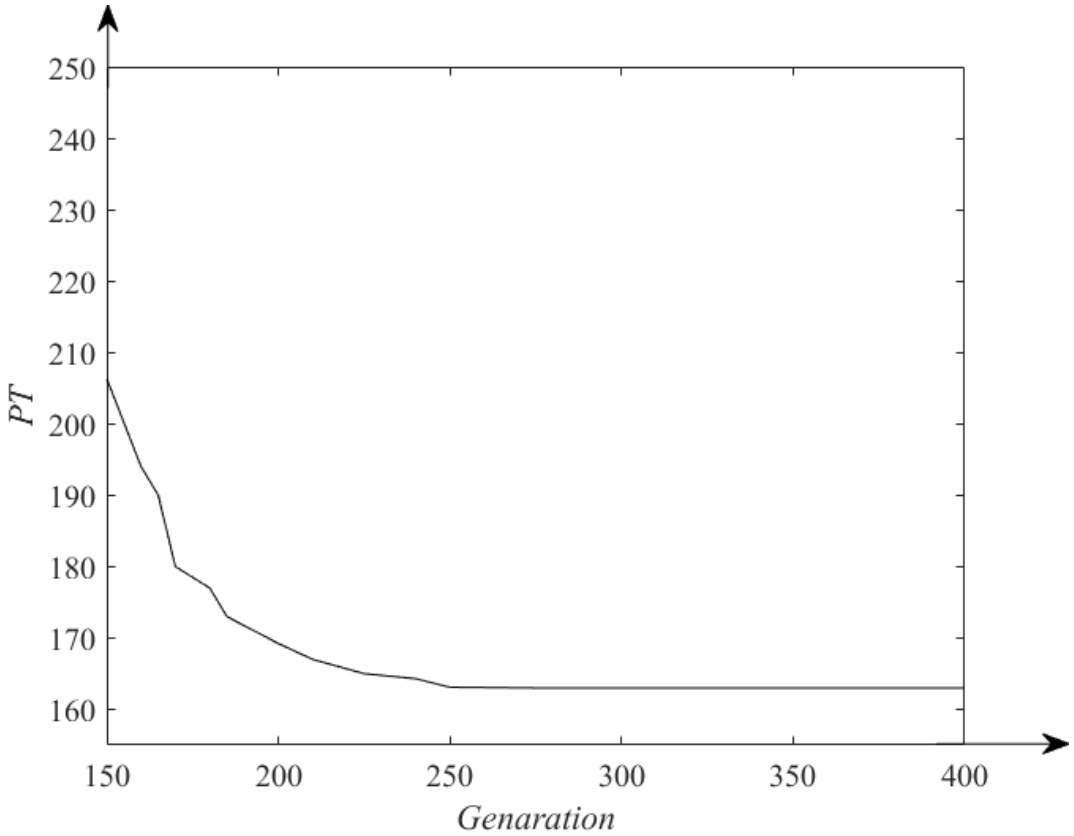


图 3.14 UNCSA (m=15,n=100) 算例迭代次数与 PT 的收敛图

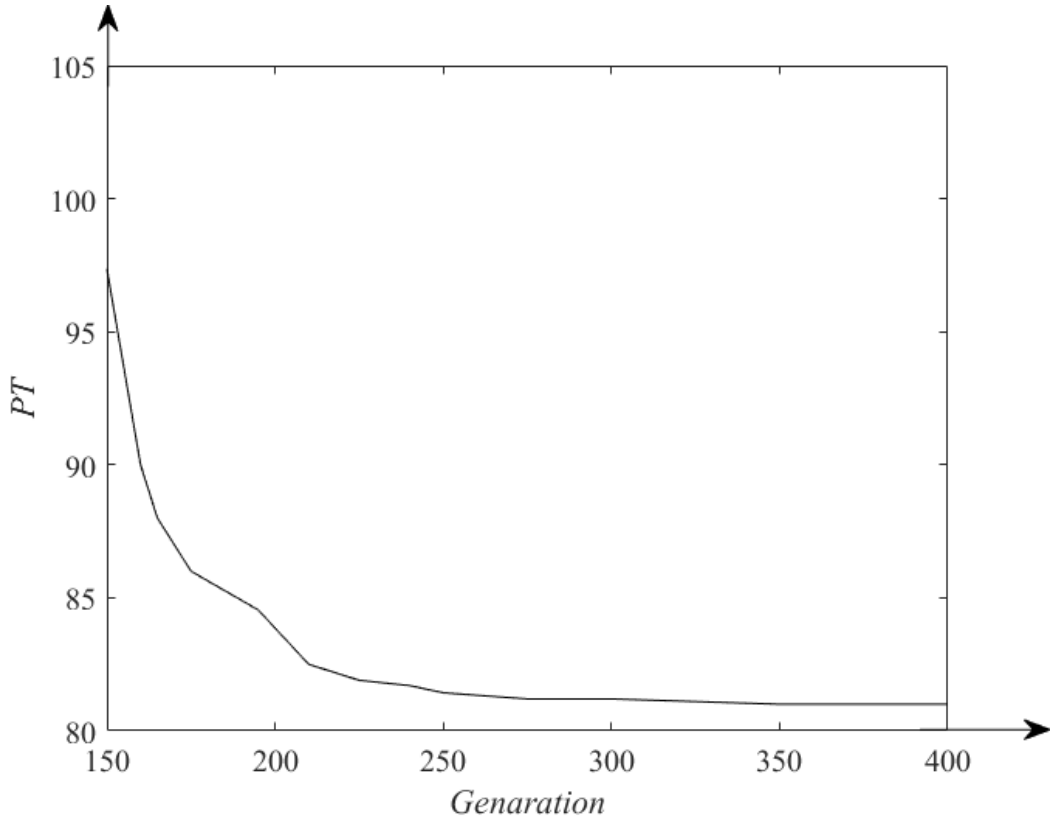


图 3.15 UNCSA (m=100,n=200) 算例迭代次数与 PT 的收敛图

3.3.4 四种算法求解对比

为了测试 UNCSA 算法在求解 PTM 模型时的有效性和优势，本文将采用 UNCSA 算法和其他三种求解一致并行机问题的智能优化算法一起进行求解对比：Laha 等[25]设计的加入了普通局部搜索的布谷鸟算法（ICSA），Lee 等[47]设计的加入了局部搜索的遗传算法（GA+LS），还有 Xu[42]等采用的模拟退火算法（SA）。其中 UNCSA 算法的参数设置如 3.3.2 节所示，并根据上一节的测试结果，最大迭代次数 $Maxgen=250$。其余算法的参数设置参考各自的文献，ICSA 中的步长因子 $\alpha=1$，$\beta=1.5$，$p_a=0.3$，最大迭代次数 $Maxgen=500$。GA+LS 中遗传算法的交叉概率为 0.8，变异概率为 0.1，进行局部搜索的概率为 0.15，终止准则为最大迭代次数 $Maxgen=300$。模拟退火算法中第 t 代的温度 $T=t/300$，当算法连续 $n(n-1)/2$ 代未发生性能改善时终止，n 为工件个数。ICSA 和 GA+LS 的种群大小 $PN=|\Lambda|=16$，所有算法的初始解均通过 LPT 规则产生。

根据 3.3.3 节的仿真结果，参数 t_β 对算法的运行趋势和规律没有影响，因此在下文中，将只对某个固定 t_β 下的实验结果进行展示分析。

表 3.5 展示了在 $t_\beta=0.8$ 时不同算法对不同规模算例的求解情况。

对于每组参数组合 $(m,n,0.8)$ 的 20 次仿真测试，选择目标函数 PT 性能值最优的那次进行展示. 对比了不同算法求解所用的 CPU 时间和最终求得的 PT 值。

表 3.5 展示了在 $t_\beta=0.8$ 时不同算法对不同规模算例的求解情况。从表中可以看出，在求解规模较小，如 $(m*n=3*5,3*8)$ 时各算法的求解结果还未看出明显的差距。但整体来看，在对各种不同规模的算例求解中，UNCSA 无论在求解时间还是求解结果上都领先于其他算法。相比于同样在布谷鸟算法中加入局部搜索的 ICSA，UNCSA 能够在更短的时间内求得更好的 PT 值。这一结果说明本文使用的面向 PTM 问题的合并场景邻域比普通邻域构造对解空间的挖掘更有效合理，大大提升了 UNCSA 的局部搜索质量，使其能在每一次的迭代过程中对解空间进行更加全面深入的搜索。作为同样采用群智能优化算法混合局部搜索的 GA+LS 算法，虽然比起 SA 算法能够求得更好的 PT，但其在四种算法中花费了最多的求解时间。而且这一时间上的差距随着问题规模的增加更加明显。这一结

果反应了 GA 和 CSA 之间求解效率之间的差距，相比 GA，CSA 能够更加快速高效地求解更为复杂的调度问题。而作为典型的串行邻域搜索算法，SA 的每次迭代过程只涉及单一解的邻域搜索，因此相比 GA+LS 消耗相对较少的 CPU 运行时间，求解时间与 ICSA 相近。但其对解空间进行的探索范围有限，在所有测试算法中最后的求解结果性能最差。

表 3.5 不同算法求解结果对比

	m	*n*	*T*	UNCSA		ICSA		GA+LS		SA	
				CPU	*PT*	*CPU*	*PT*	*CPU*	*PT*	*CPU*	*PT*
E_1	3	5	**79.94**	15.28	31.10	17.38	31.10	24.06	31.10	19.70	31.10
	3	8	**123.63**	19.26	52.09	20.74	52.09	26.90	56.52	21.69	63.98
	3	10	**171.44**	17.70	44.49	21.05	45.71	28.74	48.70	23.53	52.85
	3	15	**256.94**	22.06	66.78	25.90	69.51	34.76	80.95	28.29	88.97
	3	20	**340.25**	27.37	48.70	29.35	60.21	39.99	85.70	31.71	130.15
	5	10	**86.19**	21.21	55.55	25.08	55.55	29.68	64.50	24.52	77.46
	5	12	**119.19**	22.02	29.59	26.54	31.33	32.13	36.58	26.21	45.60
	5	15	**138.75**	25.25	65.88	30.51	74.13	41.59	80.76	33.88	84.14
	5	18	**174.50**	29.33	61.88	34.24	75.04	46.74	104.38	37.83	113.35
	5	20	**188.75**	30.36	68.89	36.89	70.96	48.63	105.00	38.99	131.32
E_2	3	50	**843.75**	46.18	165.69	54.37	182.94	73.71	228.68	56.57	596.54
	3	100	**1718.38**	84.27	229.46	98.70	367.92	129.86	433.32	99.52	1333.17
	3	200	**3374.06**	182.21	381.84	215.79	399.43	288.38	421.03	221.29	1694.95
	5	50	**515.56**	48.60	100.48	56.74	112.63	76.36	171.21	60.61	316.63
	5	100	**1035.38**	99.59	55.65	117.82	64.94	158.46	100.63	121.45	244.64
	5	200	**2063.68**	150.18	315.91	181.06	380.40	244.14	713.31	184.05	1812.97
	10	25	**129.75**	30.53	40.73	35.58	46.38	48.37	67.57	38.22	73.54
	10	50	**256.50**	58.98	46.02	72.01	56.79	91.59	88.75	71.80	108.39
	10	100	**513.50**	84.92	83.94	100.31	85.09	133.36	105.15	102.14	137.86
	15	20	**66.63**	32.79	64.36	39.61	64.36	53.05	68.38	42.45	71.56
E_3	3	500	**8454.13**	375.95	2171.35	436.55	2265.45	586.04	3719.40	435.19	7548.42
	5	500	**4945.25**	381.74	1609.65	448.06	1858.67	605.26	3754.88	445.67	7573.40
	10	200	**1020.00**	168.36	161.78	183.78	188.83	246.00	321.51	185.61	929.80
	10	500	**2537.94**	389.75	224.40	466.87	484.65	604.59	729.46	440.04	1762.20
	15	100	**341.81**	106.66	162.07	113.28	176.05	138.42	217.70	115.67	329.96
	15	200	**679.56**	117.20	277.42	130.77	340.22	152.09	403.90	133.99	570.25
	15	500	**1692.81**	402.88	224.48	454.71	585.27	598.79	809.77	436.10	1075.11
	100	200	**84.31**	190.61	82.22	249.27	95.82	336.28	120.06	256.30	181.68
	100	500	**251.46**	434.65	133.83	473.09	140.85	635.55	166.06	473.73	248.26
	100	1000	**472.12**	795.66	362.14	900.53	429.46	1207.63	518.20	894.37	760.34

为了更清楚地反应 UNCSA 相比其他算法的求解优势，本文根据 ($m*n=5*20, 10*50, 15*100$) 三个规模算例求解的 20 次结果绘制了箱型图，如图 3.16~3.18 所示。其中虚线上下的横线分别代表了结果的最大值和最小值。矩形中的红线为多次求解结果的中位数，红色十字为异常值。从图中可以看出，UNCSA 算法无论在最优解或是多次求解的平均结果上都明显优于其他三种算法，其次是 ICSA 算法和加入了局部搜索的遗传算法 GA+LS，SA 的求解能力最差，所得解与其他算法之间有着明显的差距。通过观察箱型图中解的分布情况，可以发现 UNCSA 算法所求结果分布稳定，在对同一规模算例的多次运行中求得的性能最优解和最劣解之间差距相对不大。而其他三种算法在求解时结果的上下界能看到明显的跨度，甚至在部分规模的求解中还出现了与平均求解结果差距过大的异常值（如 ICSA 算法对 $m*n=15*100$ 算例的求解，GA+LS 算法对 $m*n=5*20$ 算例的求解，SA 对 $m*n=10*50, 15*100$ 算例的求解）。这一情况表明 UNCSA 算法的搜索方式稳定有效，在多次的实验中都能保证求得相对优秀的结果。

以上的仿真结果充分体现了 UNCSA 算法相较其他算法在求解 PTM 模型时的优势。

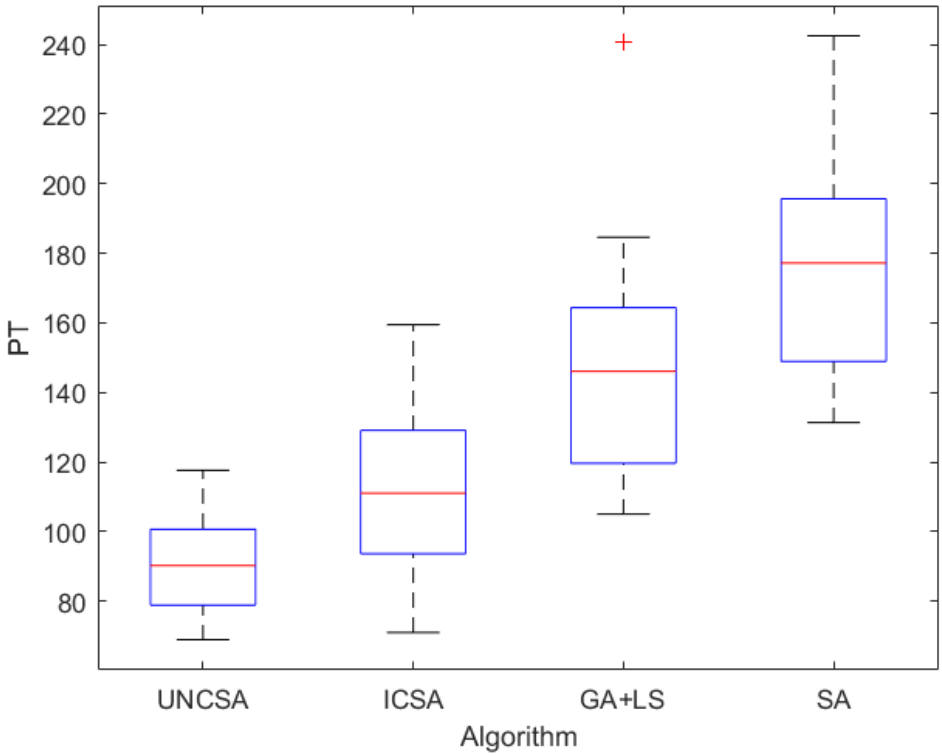


图 3.16　(m=5,n=20) 不同算法求解结果箱型图

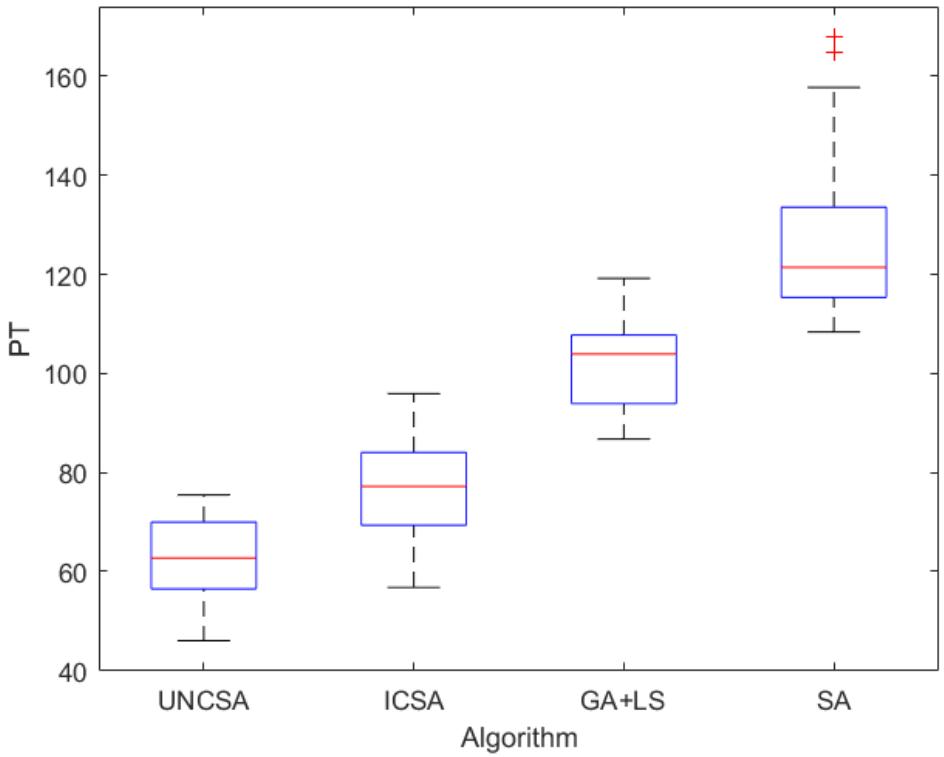


图 3.17　(m=10,n=50) 不同算法求解结果箱型图

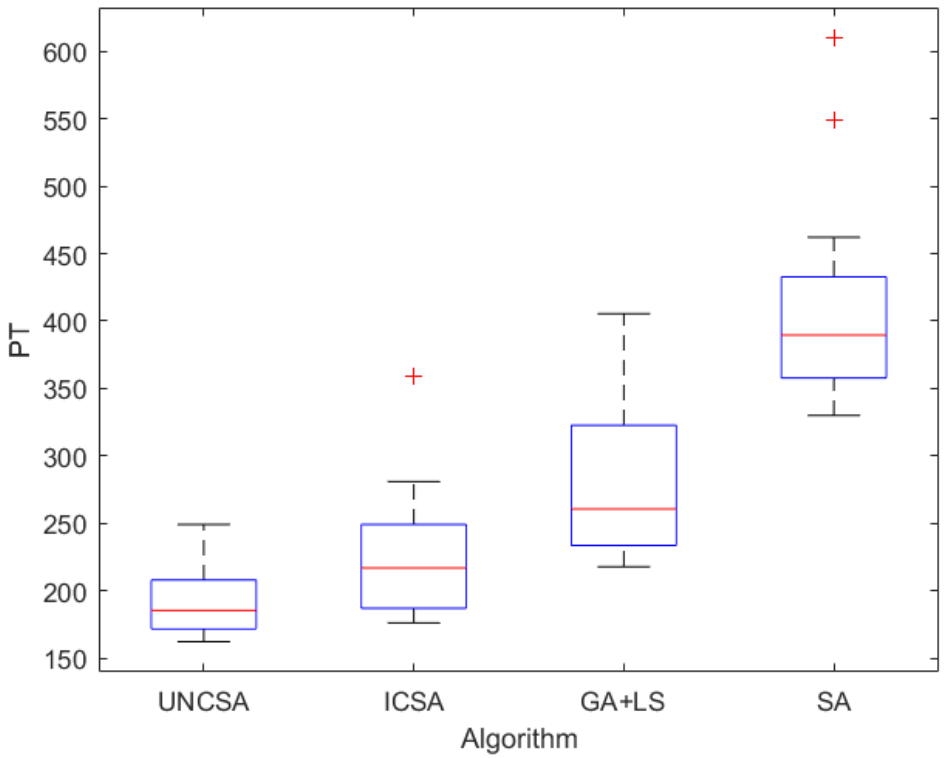


图 3.18　(m=15,n=100) 不同算法求解结果箱型图

3.4 本章小结

本章研究了给定阈值下的阈值坏场景集惩罚模型 PTM 的求解算法。为了求解该模型，本章设计了一种基于合并坏场景集邻域的混合布谷鸟算法 UNCSA。在传统的布谷鸟算法框架上加入了离散化的步骤使其能用于求解一致并行机调度问题。为了进一步提高并加入了局部搜索的步骤，将布谷鸟算法强大的全局搜索能力与局部搜索相结合，混合不同搜索方式的优点来提升算法的求解效率。同时设计了一种面向问题的合并坏场景邻域用于局部搜索，根据问题特点产生更加有针对性的邻域解来提升局部搜索的效率。

本章对所提出的 UNCSA 算法进行了大量的仿真测试，针对不同规模的一致并行机调度问题进行了求解分析。首先测试了算法求解时的终止条件，之后将 UNCSA 算法与已有的三种求解一致并行机调度问题的智能优化算法，包括加入了普通局部搜索的布谷鸟算法 ICSA，加入了普通局部搜索的遗传算法 GA+LS，还有模拟退火算法 SA 进行了求解对比。利用四种算法求解离散场景下一致并行机调度问题的 PTM 模型，通过比较求解时间和最终解的性能验证了本文设计的 UNCSA 算法在求解该模型时的优势，也表明了本文采用的基于合并坏场景的邻域构造方式的有效性和实际应用价值。

第四章　一致并行机两阶段阈值坏场景集模型及其求解算法

4.1 引言

在上一章中本文对阈值预先给定情况下的阈值坏场景集惩罚模型 PTM 进行了研究讨论，设计了一种基于合并坏场景邻域布谷鸟算法 UNCSA 进行求解，并通过仿真实验说明了该算法在求解 PTM 模型时的优势。本章将在上一章的研究基础上对整个两阶段阈值坏场景集模型 TSPTM 进行求解，首先对两阶段模型的性质进行讨论，根据模型的特点设计了一个两阶段求解框架。在第一阶段引入了均值场景的概念，再利用割平面法多场景下的最佳场景均值进行求解，作为阈值的合理取值下界。然后在第二阶段采用 UNCSA 算法求解不同合理阈值下的 $\mathrm{PTM}|T$ 模型。

本章的内容安排如下：4.2 节将对离散场景下一致并行机的两阶段阈值坏场景模型作进一步分析和讨论，提出一个代理求解框架。4.3 节是对用于求解两阶段模型的算法介绍，包括求解第一阶段的割平面法和用于求解第二阶段的混合布谷鸟算法。4.4 节是实验是和仿真，包括对第一阶段割平面法求解性能的测试以及采用两阶段算法对本章提出的两阶段模型进行仿真测试，以此来说明本章设计的两阶段求解算法以及两阶段模型的合理性。4.5 节是对本章内容的总结。

4.2 代理一致并行机两阶段阈值坏场景集模型框架

在 2.4.4 节中介绍了两阶段阈值坏场景集模型 TSPTM， 理论上来说，在合理的阈值取值范围内，TSPTM 可以生成无数组 $(T, PT(\mathrm{X}^b, T))$ 供决策者进行选择。但在实际的生产活动中，只需要根据有限数量的合理阈值生成相应的有效解即可满足决策者的不同需求。因此本文设计了一种为 TSPTM 产生有限个有效解的两阶段代理模型框架（Surrogate Two-stage PTM Framework, STPF）：

（STPF）

第一阶段

$$(\text{MCM}) \qquad MC^* = \min_{\mathrm{X}\in SX} \Big/ {1}{\big|\Lambda\big|} \sum_{\lambda\in\Lambda} C(\mathrm{X},\lambda) \tag{4.1}$$

第二阶段

$$(\text{PTM}\,|\,T) \qquad \min PT(\mathrm{X}) = \sum_{\lambda\in\Lambda_T(\mathrm{X})} \big[C(\mathrm{X},\lambda) - T\big]^2 \tag{4.2}$$

$$T = \varepsilon \cdot MC^* \tag{4.3}$$

其中满足

$$\begin{aligned} &\varepsilon = \varepsilon_1, \varepsilon_2, ..., \varepsilon_x \\ &\varepsilon_1 = 1, \varepsilon_{k+1} = \varepsilon_k + \Delta\varepsilon, \Delta\varepsilon > 0 \\ &k = 1, 2, ..., x-1 \\ &\big|\Lambda_T(\mathrm{X})\big| > 0 \end{aligned} \tag{4.4}$$

在 STPF 的第一阶段，先求解 MCM 模型，得到场景集 Λ 下的 MC^*。在第二阶段以 MC^* 为基准不断增加阈值，对一系列阈值 T 下的 $\text{PTM}\,|\,T$ 进行求解，得到一系列 $PT(\mathrm{X}^b, T)$。其中原本 TSPTM 中对 WCM 模型的求解被 $\big|\Lambda_T(\mathrm{X})\big| > 0$，即坏场景集 $\Lambda_T(\mathrm{X})$ 非空这一条件取代。根据 2.4.4 节的介绍，在 $\big|\Lambda_T(\mathrm{X})\big| > 0$ 时 $\text{PTM}\,|\,T$ 为有效的模型，必然满足阈值在合理的取值区间内，即一定满足 $T \le WC^*$。

这些不同的 $(T, PT(\mathrm{X}^b, T))$ 构成 STPF 的最终有效解集（Set of Efficient Solutions, SES）。其中的阈值取值从 MC^* 开始逐步增加，每次增加的比例步长为根据阈值 T 的合理取值区间 $[MC^*, WC^*]$ 确定的 $\Delta\varepsilon$，由决策者确定，$x = \left\lfloor \dfrac{WC^* - MC^*}{\Delta\varepsilon \cdot MC^*} \right\rfloor$，即不超过取值区间上下界长度与阈值增加量步长 $\Delta\varepsilon \cdot MC^*$ 相除的最大整数。不同阈值之间的关系用图例表示如图 4.1 所示，其中的 T_1, $T_2, ..., T_x$ 为在有效区间内，从 MC^* 开始，根据 $\Delta\varepsilon \cdot MC^*$ 步长递增选取的各阈值。理论上 STPF 可以产生 x 个两阶段阈值模型 TSPTM 的有效解。

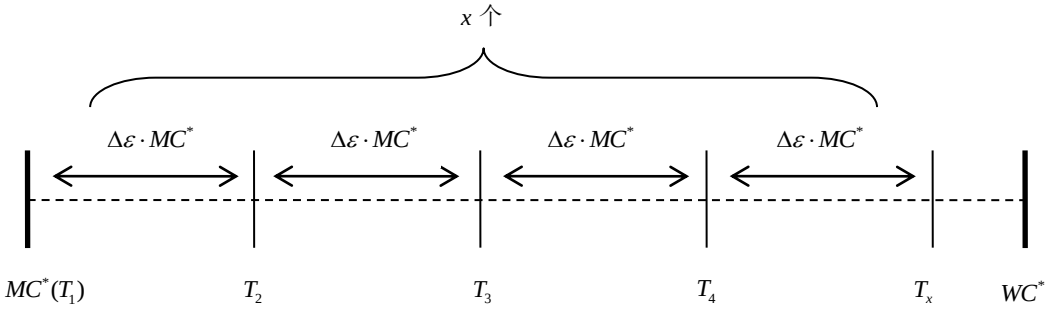


图 4.1 STPF 不同阈值之间的关系

4.3 一致并行机两阶段代理模型的求解算法

在上一节中，本文讨论了为 TSPTM 产生有限个有效解的两阶段代理模型框架 STPF，在本节中将对 STPF 的求解算法进行研究和讨论。

从式 (4.1) 和 (4.2) 可以看出，整个 STPF 的求解分为两个阶段，有着不同的目标函数和求解精度要求。第一阶段是对离散场景下 MCM 模型的求解，所得的结果 MC^* 将作为第二阶段的阈值初始值，因此需要精确的求解结果来保证阈值的合理性，选择精确算法来求解。而第二阶段的 PTM$|T$ 目标函数是所有坏场景下的惩罚值，目标函数相对复杂，并涉及多组阈值下的目标函数求解。因此在第二阶段将选用求解效率更高的启发式智能优化算法进行求解。整个用于 STPF 的两阶段求解算法的求解框架如图 4.2 所示。

用于第一阶段求解的算法记作 A1（Algorithm 1），用于第二阶段求解的算法记作 A2（Algorithm 2）。在第一阶段利用 A1 求解输入的场景集 Λ 下的 MCM 模

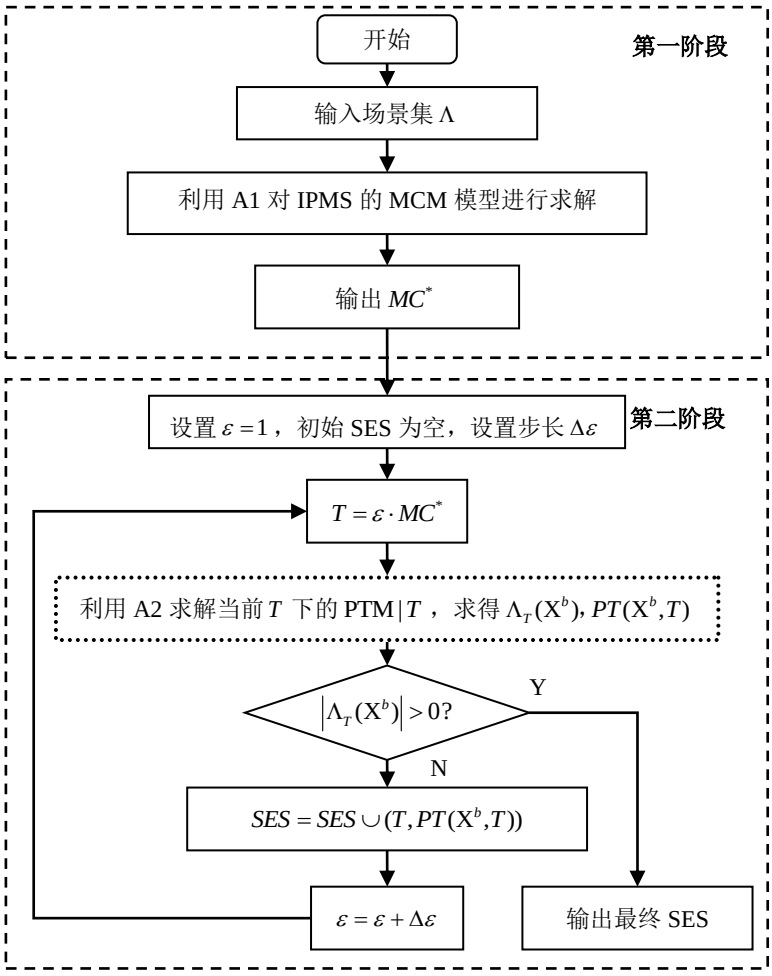


图 4.2 两阶段算法求解框架

型，将计算得到的MC^*传入第二阶段。在第二阶段中将T从MC^*开始以$\Delta\varepsilon$的步长逐渐增加，在每个阈值T下利用 A2 进行 PTM$|T$的求解。并将得到的$(T, PT(\mathrm{X}^b, T))$加入最终有效解集 SES。

据本文了解，在一致并行机调度中尚未有直接用于求解一致并行机 MCM 的精确算法，因此本文提出了一种均值场景的概念，将场景集Λ下的一致并行机 MCM 问题转换为均值场景下的确定性一致并行机调度问题 IPMS，然后选用原本用于 IPMS 问题的割平面法[11]作为求解第一阶段的 A1。

而在第二阶段选择了近似算法进行求解，因此可能会出现伪有效解。即当前的阈值T已经超过了合理的取值范围，但近似算法求解的不彻底导致依旧可以求得$(T, PT(\mathrm{X}^b, T))$的情况。这也是采用近似算法代替精算法的成本。所以需要尽可能选择求解能力更强的启发式算法。在第三章中本文设计了一种 UNCSA 对给定阈值下的 PTM 模型进行求解，并通过实验证明了该算法的有效性，而第二阶段问题的求解可以看作对多个阈值下的 PTM 的求解，因此本文选用 UNCSA 算法作为求解第二阶段的 A2。

4.3.1 求解第一阶段的割平面法

在采用割平面法对第一阶段的 MCM 进行求解时，首先要进行模型的转换。对于离散场景下一致并行机的 MCM 模型，其可以转换成单一场景下的确定性一致并行机调度问题 IPMS 进行求解。

定义 4.1 以每个工件的加工时间为该工件在Λ中所有场景下加工时间的均值所构造的场景称为场景集Λ下的均值场景，用$\bar{\lambda}(\Lambda)$表示，简记为$\bar{\lambda}$。则$p_i^{\bar{\lambda}} = \frac{1}{|\Lambda|}(p_i^{\lambda_1} + p_i^{\lambda_2} + ... + p_i^{\lambda_{|\Lambda|}})$ $i = 1, 2, ..., n$， $\lambda_i \in \Lambda$。

定理 1 对于场景集Λ下的不确定一致并行机调度问题，其 MCM 模型可以转换为均值场景$\bar{\lambda}$下的 IPMS 模型的形式。

证明：用y_i表示第$i(i = 1, 2, ..., |\Lambda|)$个场景下的 makespan,则场景集$\Lambda$下的一致并行机 MCM 模型的数学规划形式表达如下：

$$\min \quad y = \frac{1}{|\Lambda|}\sum_{i=1}^{|\Lambda|} y_i \tag{4.5}$$

$$s.t\ \sum_{j=1}^{m} x_{ij} = 1,\ \ 1 \le i \le n, \tag{4.6}$$

$$y_1 - \sum_{i=1}^{n} p_i^{\lambda_1} x_{ij} \ge 0,\ \ 1 \le j \le m, \tag{4.7}$$

$$y_2 - \sum_{i=1}^{n} p_i^{\lambda_2} x_{ij} \ge 0,\ \ 1 \le j \le m, \tag{4.8}$$

$$\cdots$$

$$y_{|\Lambda|} - \sum_{i=1}^{n} p_i^{\lambda_{|\Lambda|}} x_{ij} \ge 0,\quad 1 \le j \le m, \tag{4.9}$$

$$x_{ij} \in \{0,1\} \tag{4.10}$$

将(4.7)~(4.9)相加，表示为：

$$|\Lambda| y - [\sum_{i=1}^{n} (p_i^{\lambda_1} + p_i^{\lambda_2} + ... + p_i^{\lambda_{|\Lambda|}}) x_{ij}] \ge 0,\ \ 1 \le j \le m, \tag{4.11}$$

对(4.11)作进一步简化：

$$y - [\sum_{i=1}^{n} \frac{1}{|\Lambda|} (p_i^{\lambda_1} + p_i^{\lambda_2} + ... + p_i^{\lambda_{|\Lambda|}}) x_{ij}] \ge 0,\ \ 1 \le j \le m, \tag{4.12}$$

根据均值场景 $\bar{\lambda}$ 的定义，均值场景下的各工件加工时间 $p_i^{\bar{\lambda}} = \frac{1}{|\Lambda|}(p_i^{\lambda_1} + p_i^{\lambda_2} + ... + p_i^{\lambda_{|\Lambda|}})$，则离散场景集 Λ 下的 MCM 模型，可以转化为求均值场景 $\bar{\lambda}$ 下的 IPMS 的表达形式：

$$\min\ \ y \tag{4.13}$$

$$s.t\ \ \sum_{j=1}^{m} x_{ij} = 1,\ \ 1 \le i \le n, \tag{4.14}$$

$$y - \sum_{i=1}^{n} p_i^{\bar{\lambda}} x_{ij} \ge 0,\ \ 1 \le j \le m, \tag{4.15}$$

$$x_{ij} \in \{0,1\} \tag{4.16}$$

□

根据以上的结论，MCM 问题可以看作单一场景 $\bar{\lambda}$ 下的 $P_m \| C_{\max}$，通过已有的求解 IPMS 的方法求解(4.13)~(4.16)，将求得的最优解 X^* 代入 MCM 的计算公式(2.9)中，即可得到所有场景下的最佳均值性能 MC^*。

因此在对第一阶段进行求解时，首先根据定义 4.1 生成场景集 Λ 下的均值场景 $\bar{\lambda}$，均值场景中的每个工件的加工时间为原场景集 Λ 中所有场景 λ 下的加工时间的均值，即 $p_i^{\bar{\lambda}} = \frac{1}{|\Lambda|}(p_i^{\lambda_1} + p_i^{\lambda_2} + ... + p_i^{\lambda_{|\Lambda|}})\ \ i = 1,2,...,n; \lambda \in \Lambda$，然后将均值场景 $\bar{\lambda}$ 下的

各工件加工时间输入到 2.5 节中介绍的割平面法中进行求解。整个用于第一阶段的算法 A1 的伪代码如下图所示：

Algorithm 1：求解第一阶段的割平面法

Input: 场景集 Λ
Output: Λ 下的最优场景均值 MC^*
Begin:
Step1：计算 Λ 下的均值场景 $\bar{\lambda}$，其中 $p_i^{\bar{\lambda}} = \frac{1}{|\Lambda|}(p_i^{\lambda_1} + p_i^{\lambda_2} + ... + p_i^{\lambda_{|\Lambda|}})\ \ i = 1,2,...,n; \lambda \in \Lambda$
Step2：采用 2.5 节中的割平面法对 $\bar{\lambda}$ 下的 IPMS 进行求解，得到最优解 X^*
Step3：将 X^* 代入 Λ 下的 MCM 模型中，求得 MC^*
Step4：输出最优场景均值 MC^*
End

图 4.3 求解第一阶段的 A1 算法伪代码

4.3.2 求解第二阶段的混合布谷鸟算法

在第三章中本文对给定确定阈值的 IPMS 下的 PTM 问题进行了求解。根据问题的特点，本文在第三章设计了一种面向问题的合并坏场景集邻域构造方式，并将采用了这种邻域构造方式的局部搜索与布谷鸟算法相结合，形成了基于坏场景集邻域的混合布谷鸟算法 UNCSA 用于 PTM 模型的求解。第三章大量的实验仿真结果表明相比已有的智能优化算法，该算法可以高效地对给定阈值下的 PTM 问题进行求解。

在本章中，对 STPF 第二阶段问题的求解可以看作是对多组确定阈值 T 下的 PTM 模型的求解。因此将第三章中设计的 UNCSA 算法作为本章第二阶段问题的求解算法完全适合。第二阶段的求解过程可以看作由内外两层迭代组成，外层的迭代为 T 值的更新，每运行一次就增加一次当前阈值与 MC^* 的比例 ε，比例增加的步长为 $\Delta\varepsilon$。内层的迭代则为 UNCSA 算法本身的迭代过程。内层迭代的终止条件为最大迭代次数 $Maxgen$，外层迭代的终止条件则为内层 UNCSA 求解的当前阈值下的坏场景集 $\Lambda_T(X^b)$ 为空。

用于第二阶段求解的算法 A2 的伪代码如下所示：

Algorithm 2：求解第二阶段的 UNCSA

Input: Λ 下的最优场景均值 MC^*，阈值比例递增步长 $\Delta\varepsilon$
Output: 不同阈值 T 下求得的 $(T, PT(\mathrm{X}^b, T))$ 组成的有效解集 SES
Begin:
Step1：设置初始阈值比 $\varepsilon = 1$，清空最终有效解集 SES。
Step2：$T = \varepsilon \cdot MC^*$，利用 UNCSA 求解当前阈值 T 下的 $\mathrm{PTM}|T$，求得 $\Lambda_T(\mathrm{X}^b), PT(\mathrm{X}^b, T)$
Step3：判断 $\Lambda_T(\mathrm{X}^b)$ 是否为空。如果是，则跳转，如果 $\Lambda_T(\mathrm{X}^b)$ 非空，则继续 Step4。
Step4：将当前阈值 T 求得的 $(T, PT(\mathrm{X}^b, T))$ 加入 SES，并使 $\varepsilon = \varepsilon + \Delta\varepsilon$，跳转 Step2
Step5：输出 SES。
End

图 4.4 求解第二阶段的 A2 算法伪代码

4.4 仿真实验与分析

在本节中将采用上一节设计的两阶段算法对 SIPMS 中的 STPF 进行求解。问题的算例设置与第三章相同。在本节中将首先对用于第一阶段的割平面法进行性能测试，看能否对不同规模的算例都进行有效的求解。然后将采用 UNCSA 算法求解第二阶段的问题，并与其他的智能优化算法进行对比，显示算法求解精度不同造成的伪有效解影响。最后通过实验仿真来展示 STPF 模型的一些性质。

本文中的所有仿真程序均采用 C++语言编写实现，在 Windows10 操作系统环境下采用 Visual Studio 2017 和 CPLEX 12.8.0 软件运行。仿真所用计算机的 CPU 为 Intel（R）Xeon（R）Gold 5188 ,运行内存为 256G。

4.4.1 第一阶段割平面法的性能测试

根据上文的讨论，第一阶段中将先把场景集 Λ 下的一致并行机调度问题的 MCM 模型转换为均值场景 $\bar{\lambda}$ 下的确定性一致并行机调度问题 IPMS，然后采用 Mokotoff[11]设计的割平面法进行求解。本文中设定合理的求解时间为 $5000s$，即当算法求解超过 $5000s$ 的时间则认为此时为无效求解。

在 3.3.3 节的仿真中我们发现，对于 UNCSA 算法，算例的加工时间参数 t_β 算例规模只是通过影响加工时间的取值分布来影响最终的目标函数值，但是对算法的运行趋势和规律并未产生明显的影响。为了探究 t_β 对割平面法的影响，本文先从 $E_1 \sim E_3$ 三种规模类型中各选取两个规模的算例，包括 E_1 中的

$(m*n=3*20,\ 5*20)$，E_2 中的 $(m*n=5*50,\ 10*50)$，E_3 中的 $(m*n=15*100,\ 100*200)$。观察六种规模的算例在场景数 $|\Lambda|=16$ 时不同时间参数 t_β 下的求解情况。测试结果如表 4.1 所示，其中的 *CPU* 为运行时间，*iterations* 为割平面法在求解该算例时执行的循环次数，MC^* 为使用割平面法求解 MCM 模型得到的最优场景均值。

从表 4.1 中可以看出，在算例规模 $(m*n)$ 固定的情况下，无论是对于 E_1 中的较小规模算例（如 $m*n=3*20$），还是对于 E_3 中的大规模算例（如 $m*n=100*200$），时间参数 t_β 产生的主要影响是该变加工时间的取值，从而影响最后求得的 MC^*。

表 4.1 不同时间参数 t_β 下的割平面法仿真结果

	m	*n*	t_β	*CPU*	*iterations*	MC^*
E_1	3	20	0.4	0.20	41	193.31
			0.6	0.24	44	267.69
			0.8	0.24	45	340.25
			1	0.28	49	422.25
	5	20	0.4	0.20	54	107.81
			0.6	0.22	59	145.38
			0.8	0.26	59	188.75
			1	0.30	63	246.25
E_2	5	50	0.4	0.62	173	295.75
			0.6	0.60	174	387.56
			0.8	0.65	179	515.56
			1	0.70	283	623.50
	10	50	0.4	1.20	317	139.13
			0.6	1.22	319	202.94
			0.8	1.25	357	256.50
			1	1.27	372	309.13
E_3	15	100	0.4	4.00	1168	191.94
			0.6	4.28	1155	269.50
			0.8	5.40	1279	341.81
			1	5.42	1482	407.44
	100	200	0.4	100.65	10193	42.19
			0.6	100.82	10202	63.25
			0.8	102.00	10415	84.31
			1	106.80	10461	105.63

当然t_β对 CPU 运算时间和迭代次数也存在一定影响。对于同一规模的算例，t_β的增大会导致加工时间取值分布空间范围增大，从而导致计算量增加，CPU 时间和迭代次数也相应的增加，规模越大时增加的绝对变化量也越大。但这一变化量与求解消耗的 CPU 运行时间和迭代次数相比所占比重过于细微，远不如不同运算规模造成的变化明显，在实际求解时可以忽略不计。因此在接下来的仿真结果展示时，本文将只选择展示$t_\beta = 0.8$的结果。

表 4.2 是不同规模的算例在时间参数$t_\beta = 0.8$时用割平面法进行求解的仿真结果。从表中可以看到，割平面法求解 MCM 模型需要的迭代次数和 CPU 时间与问题规模$m*n$有着密切的联系。CPU 时间消耗一般随着问题规模的增加而增加，这一变化趋势在大规模的算例中表现得更为明显。观察E_3中的算例可以发现，在问题规模增加时，迭代次数和 CPU 时间呈大幅度的增加趋势。如$(100*500)$规模的算例为$(100*200)$规模的 2.5 倍，但求解时间增加到了 20 倍左右。同时工件数和机器数的比值n/m对求解时间和迭代次数也有着较大的影响。越大的n/m意味着在一台机器上平均会安排更多的工件进行加工，单次迭代过程中的运算量增加，整体的运行时间也会增加。因此在规模相近甚至规模略小的情况下，n/m更大的算例会表现出更多的求解时间。以E_2中两个规模$(3*100,\ 15*20)$的算例为例，两者规模相同，工件数和机器数的乘积均为 300，但是$m*n=3*100$的算例中$n/m=100/3=33.3$，远大于$m*n=15*20$时工件数和机器数的比值 1.33。因此割平面法在求解$m*n=3*100$算例时消耗了更多的 CPU 时间。而像E_2中$m*n=3*50$的算例，虽然规模小于$m*n=15*20$的算例，但是其工件数和机器数的比值$n/m=50/3=16.7>1.33$，因此也消耗了更多的 CPU 时间。

虽然割平面法在规模增大时需要消耗一定的计算机资源，但是仿真实验结果表明在本文设定的可接受范围内割平面法可以对设定的算例进行有效的求解，可以在实际问题中进行应用，本阶段求解结果也将作为第二段中阈值的初始值使用。

表 4.2 不同规模下割平面法仿真结果

	m	*n*	*CPU*	*iterations*	MC^*
E_1	3	5	0.05	4	79.94
	3	8	0.06	10	123.63
	3	10	0.10	21	171.44
	3	15	0.12	35	256.94
	3	20	0.24	45	340.25
	5	10	0.12	16	86.19
	5	12	0.15	39	119.19
	5	15	0.18	52	138.75
	5	18	0.23	59	174.50
	5	20	0.26	59	188.75
E_2	3	50	0.62	114	843.75
	3	100	2.60	232	1718.38
	3	200	3.76	463	3374.06
	5	50	0.65	179	515.56
	5	100	3.54	358	1035.38
	5	200	4.60	720	2063.68
	10	25	0.63	154	129.75
	10	50	2.75	357	256.50
	10	100	4.50	472	513.50
	15	20	0.52	153	66.63
E_3	3	500	36.00	4162	8454.13
	5	500	42.80	4809	4945.25
	10	200	7.80	1349	1020.00
	10	500	114.40	6651	2537.94
	15	100	5.40	1279	341.81
	15	200	12.96	4886	679.56
	15	500	67.40	7296	1692.81
	100	200	102.00	10415	84.31
	100	500	2555.00	59623	251.46
	100	1000	4907.00	105128	472.12

4.4.2 第二阶段仿真测试

在这一小节中将采用第三章设计的 UNCSA 算法和其他三种求解一致并行机问题的智能优化算法一起对 STPF 的第二阶段进行求解对比：Laha 等[25]设计的加入了普通局部搜索的布谷鸟算法（ICSA），Lee 等[47]设计的加入了局部搜索的遗传算法（GA+LS），还有 Xu[42]等采用的模拟退火算法（SA）。不同算法的参数设置与 3.3.4 节中相同。将第一阶段求得的 MC^* 作为初始值，以 $\Delta\varepsilon = 0.01$ 不断

增加阈值T。比较不同算法在不同阈值下求解 STPF 第二阶段得到的一系列结果。

表 4.3 是四种算法求解$(m*n=5*20,\ 15*50,\ 15*20)$三种不同规模算例在时间参数$t_\beta=0.8$时的第二阶段的结果，表中的$\varepsilon$为当前阈值和$MC^*$的比例系数，

表 4.3 不同算法求解第二阶段结果比较

m	n	ε	UNCSA		ICSA		GA+LS		SA	
			CPU	*PT*	*CPU*	*PT*	*CPU*	*PT*	*CPU*	*PT*
5	20	1	30.36	68.89	36.89	70.96	48.63	105.00	38.99	131.32
		1.01	30.18	52.32	36.10	60.68	48.04	74.39	38.38	99.78
		1.02	29.69	43.01	36.10	52.80	48.01	67.52	38.32	77.20
		1.03	29.45	36.59	36.00	41.33	47.97	61.05	38.30	69.31
		1.04	29.44	25.34	35.90	34.76	47.94	55.90	38.27	58.24
		1.05	29.39	20.35	35.80	30.59	47.93	44.48	38.27	51.87
		1.06	29.25	16.14	35.80	28.87	47.93	37.57	38.26	46.68
		1.07	29.19	13.25	35.70	19.11	47.92	27.08	38.20	39.40
		1.08	28.74	5.51	35.70	13.56	47.88	19.48	38.13	25.17
		1.09	28.63	1.91	35.50	6.75	47.77	9.52	38.10	11.10
		1.1	28.2	0	35.2	0	47.54	0	37.9	0
10	50	1	58.98	46.02	72.01	56.79	91.59	86.75	71.80	108.38
		1.01	57.37	36.02	71.99	51.49	91.33	82.17	71.02	99.19
		1.02	57.37	31.76	71.01	46.39	90.75	63.74	70.85	86.76
		1.03	56.14	25.57	70.21	41.01	90.29	58.98	70.80	68.17
		1.04	56.10	22.04	70.08	37.86	90.06	50.53	70.74	59.92
		1.05	55.59	18.78	70.07	35.94	89.98	44.26	70.74	49.61
		1.06	55.59	16.90	70.00	28.82	89.88	32.07	70.70	37.25
		1.07	54.94	10.50	69.86	22.89	89.83	25.40	70.65	30.74
		1.08	54.72	5.80	69.85	14.62	89.77	20.04	70.53	24.82
		1.09	54.48	1.11	69.84	8.61	89.76	13.10	70.40	17.53
		1.1	54.11	0	69.76	2.09	89.72	7.73	70.34	12.13
		1.11	—	—	69.44	0	88.86	0	70.15	4.65
		1.12	—	—	—	—	—	—	69.87	0
15	100	1	85.33	162.07	103.28	176.05	138.42	217.70	105.67	329.96
		1.01	84.77	114.09	103.26	124.56	138.31	169.93	105.19	267.96
		1.02	84.35	70.28	103.13	87.49	138.26	111.48	104.85	169.98
		1.03	84.13	48.69	103.09	60.07	137.82	84.83	104.77	104.48
		1.04	83.65	38.72	102.79	47.36	137.62	61.95	104.74	89.30
		1.05	83.55	16.07	102.48	25.68	137.49	49.09	104.63	63.11
		1.06	82.76	5.38	102.17	20.08	137.27	32.27	103.92	50.80
		1.07	82.71	0	101.83	6.30	137.05	10.21	103.61	21.34
		1.08	—	—	101.70	0	137.02	0	103.54	8.77
		1.09	—	—	—	—	—	—	103.27	0

CPU 为算法求解消耗的 CPU 时间，PT 为当前阈值下求解 $\mathrm{PTM}|T$ 模型的结果。

从表中可以看出，在规模不变的情况下，阈值增加导致每次迭代过程中筛选出的坏场景数也变少，相应的会减少一定的求解时间，对应的最终 PT 值也不断减少。这一现象与 2.4.4 节中在第二阶段同一阈值下的 $\mathrm{PTM}|T$ 问题的求解中，UNCSA 的求解结果更占优势。相比其他算法，UNCSA 在消耗更少 CPU 时间的情况可以获得更好的结果。

进一步观察分析表中的数据可以发现算法求解精度对 STPF 的最终解集造成的影响。在某些情况下不同算法进行第二阶段求解时得到的最终解集 SES 中的数量不同，如在 $(m*n=10*50)$ 时 UNCSA 算法求得的 SES 中 $(T,PT(\mathrm{X}^b,T))$ 为 10 个，有效阈值从 $T=MC^*$ 增长到了 $T=1.09MC^*$，当 $T=1.1MC^*$ 时 UNCSA 求得场景集个数已经为 0，即此时对于 UNCSA 算法第二阶段外循环的终止条件为 $T=1.1MC^*$。而利用 SA 算法进行第二阶段求解的外循环终止条件为 $T=1.12MC^*$，SA 算法求得的 SES 中包含 12 个有效解。而实际上 SA 算法在 $T=1.1MC^*$ 和 $T=1.11MC^*$ 时求得的结果是伪有效解。根据 UNCSA 的求解结果，$1.1MC^*$ 和 $1.11MC^*$ 并非有效的阈值取值。产生这种现象的原因是因为第二阶段采用的算法并非精确算法，不同算法在同一阈值下得到的 PT 目标最优值存在差异。当 UNCSA 算法求得的 PT 最优解为 0，SA 求得的最优解依然大于 0。此时系统会将其判定为有效解并加入 SES 中，同时根据外循环的条件继续增加阈值进行求解，而实际上此时的阈值已经超过了合理的取值范围。假设 T' 为某一算法求解 STPF 终止时的阈值，而 T'' 为另一种算法求解的终止阈值，它们与理论的阈值上限之间可能会存在这样的关系，T'' 即为伪合理阈值，求得的为伪有效解。因此为了减少产生伪有效解，需要选用尽可能精度的算法进行求解。从表 4.3 的实验结果可以看出，根据 UNCSA 的计算结果可以使外循环终止时阈值尽可能地接近理论上阈值的合理上界。

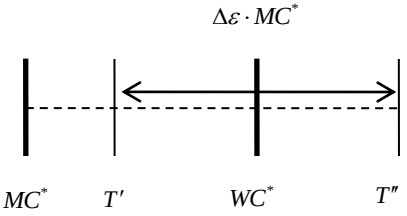


图 4.5 不同算法求解时的终止阈值与理论上限的关系图

为了更加详细地展示不同算法对第二阶段问题求解的差异，我们将不同算法求解这三个算例得到的 SES 绘制成了前沿图，如图 4.6~图 4.8 所示。从图中可以看出 UNCSA 算法在求解第二阶段的 PTM|T 模型时更占优其他三种算法。相比其他三种算法利用 UNCSA 算法可以减少伪有效解的产生。

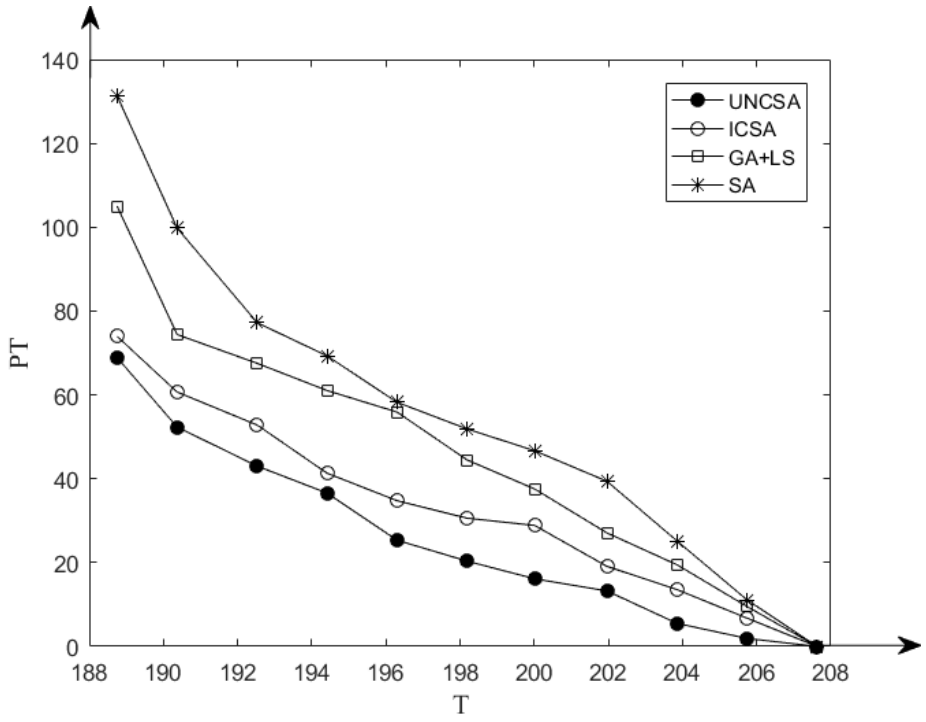


图 4.6　(m=5,n=20) 不同算法求解第二阶段前沿图

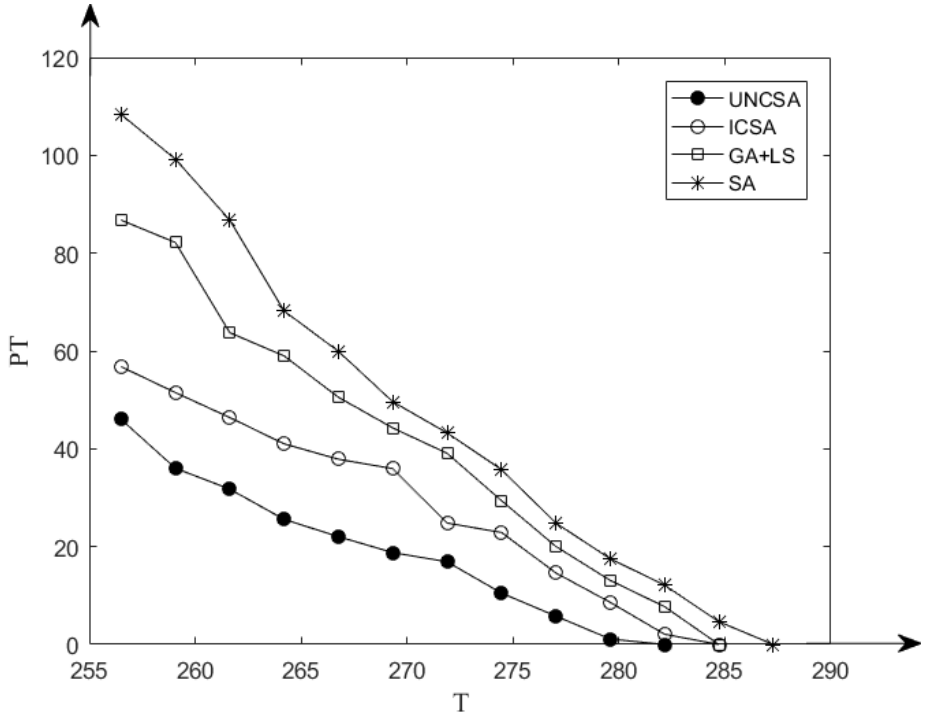


图 4.7　(m=10,n=50) 不同算法求解第二阶段前沿图

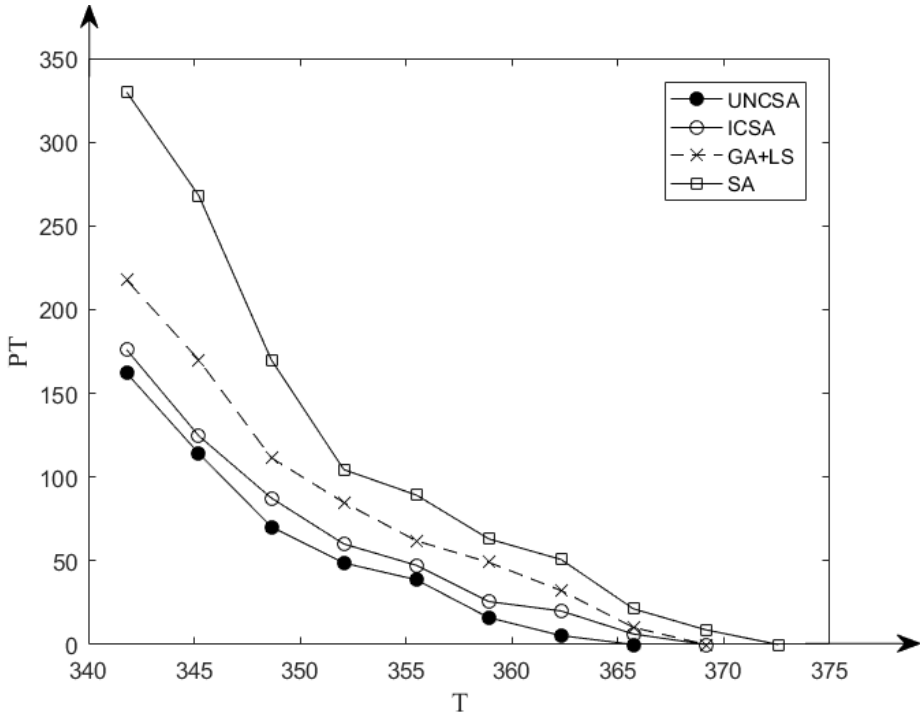


图 4.8　(m=15,n=100) 不同算法求解第二阶段前沿图

4.4.1 节和 4.4.2 节的实验仿真可以说明，利用本文设计的将割平面法和 UNCSA 算法结合的两阶段算法可以对 STPF 进行有效的求解。第一阶段的求解本文选用了精确算法，而在第二阶段选用更优秀的 UNCSA 算法可以提高求解精度，从而减少伪有效解的产生。

4.4.3　STPF 模型性质的仿真测试

在 2.4.4 中本文对两阶段阈值坏场景集模型 TSPTM 进行了讨论分析，该模型涉及两个优化目标，阈值 T 和惩罚值 $PT(\mathrm{X}^b,T)$，较小的阈值 T 可以抑制更多坏场景的性能，从而利于获得更好的期望性能，改善系统的优化性。但相对的较小的阈值 T 会导致 $PT(\mathrm{X}^b,T)$ 增加，导致系统的鲁棒性降低。而作为 TSPTM 的代替求解框架，STPF 涉及的鲁棒性和优化性也应该体现出这种性质。

本文从 $E_1 \sim E_3$ 三种规模类型中各选取两个规模的算例，包括 E_1 中的 $(m*n=3*20,\ 5*20)$，E_2 中的 $(m*n=5*50,\ 10*50)$，E_3 中的 $(m*n=15*100,\ 100*200)$，用本文设计的两阶段算法进行求解。将不同合理阈值

及该阈值下求得的最优解 X^b 对应的 $PT(X^b,T)$ 和 $MC(X^b)$ 绘制成了三维坐标图，如图 4.9~图 4.14 所示。

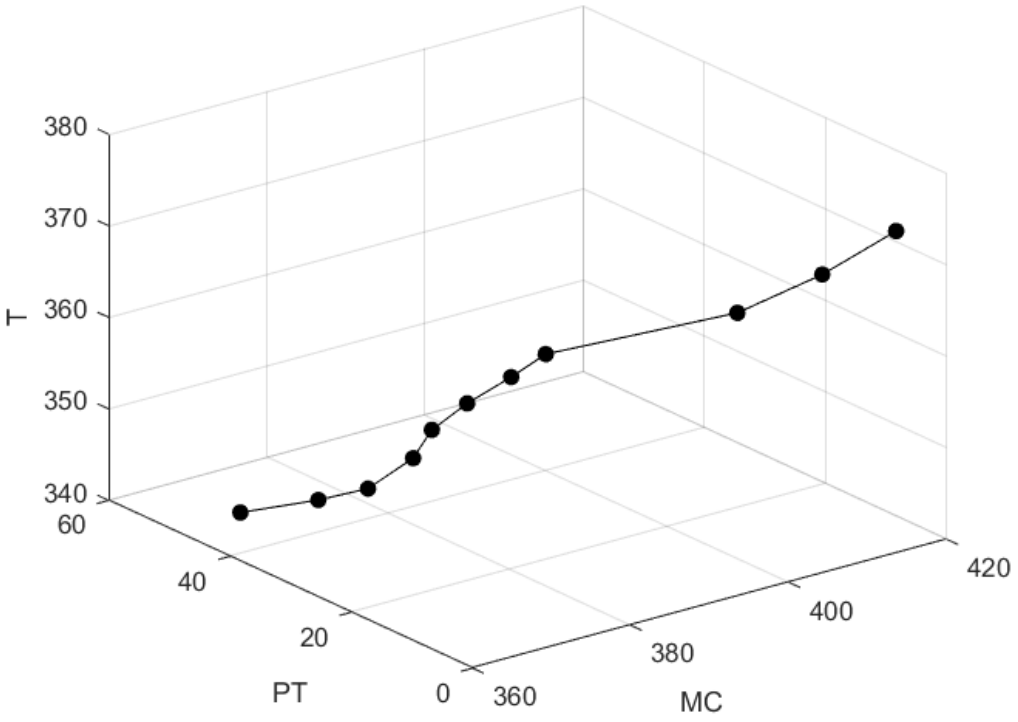


图 4.9 (m=3,n=20) STPF 不同阈值下最优解的 PT 值和场景均值

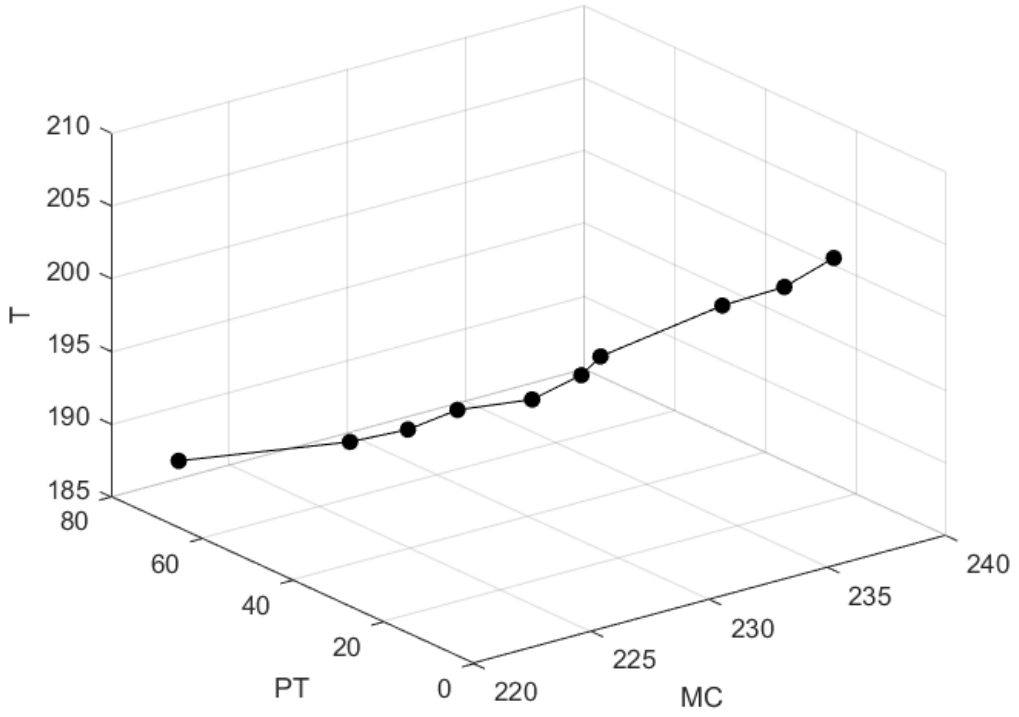


图 4.10 (m=5,n=20) STPF 不同阈值下最优解的 PT 值和场景均值

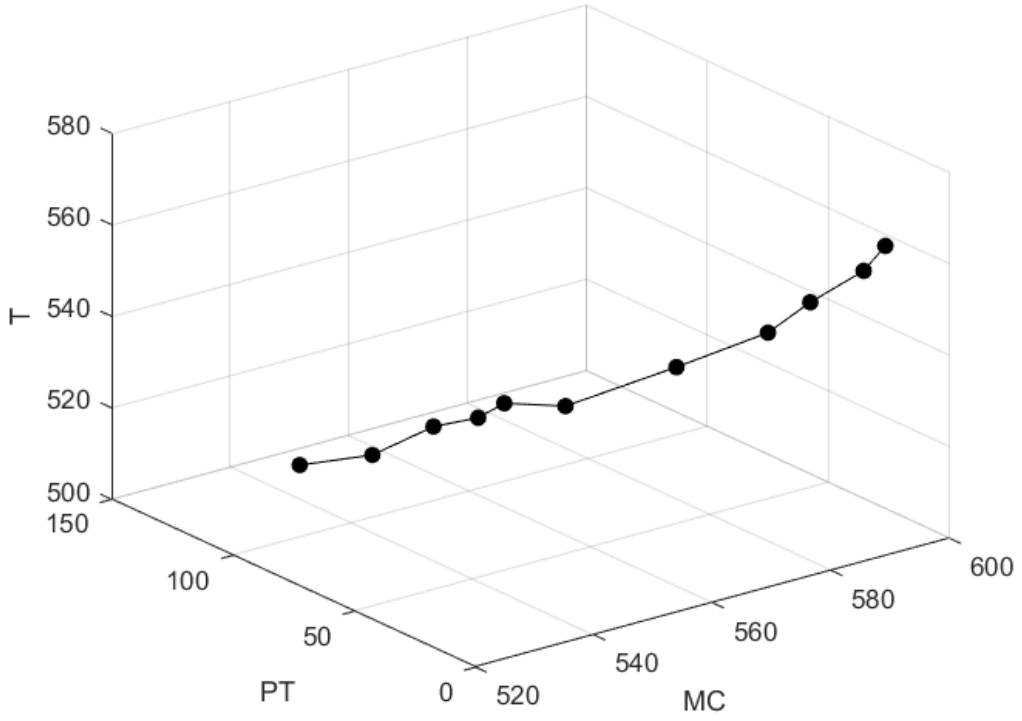


图 4.11 (m=5,n=50) STPF 不同阈值下最优解的 PT 值和场景均值

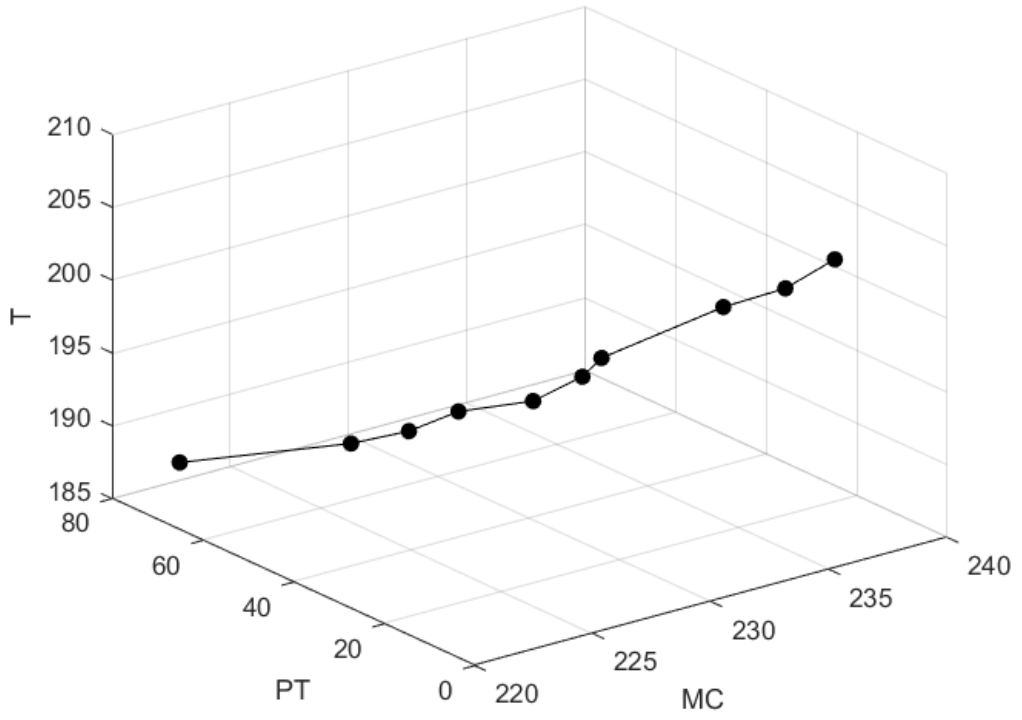


图 4.12 (m=10,n=50) STPF 不同阈值下最优解的 PT 值和场景均值

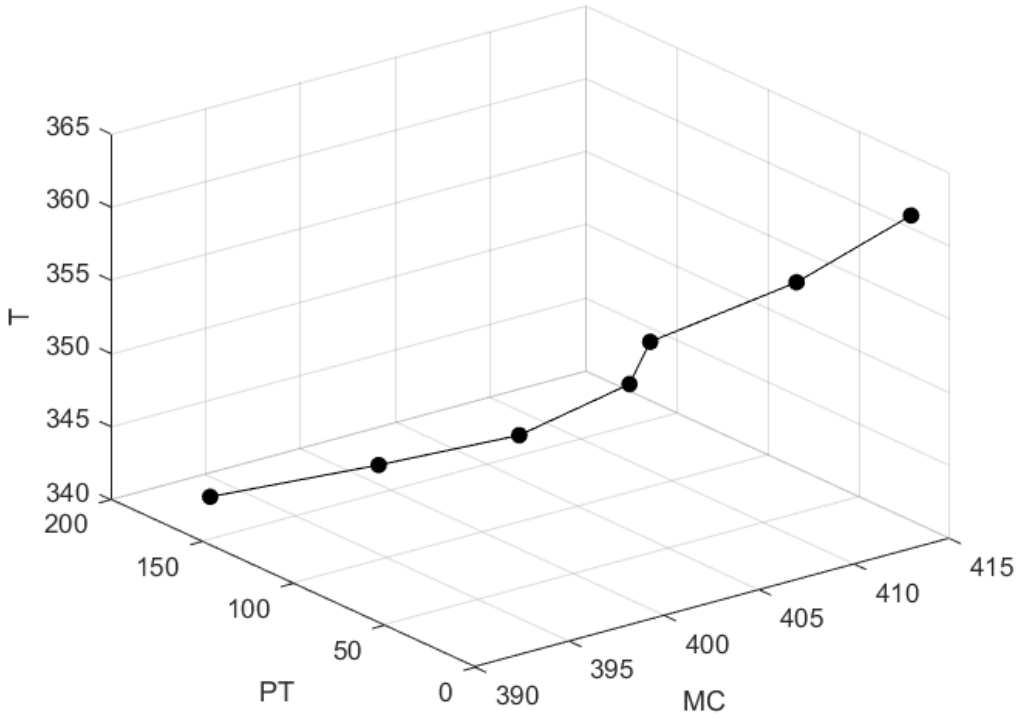


图 4.13 (m=15,n=100) STPF 不同阈值下最优解的 PT 值和场景均值

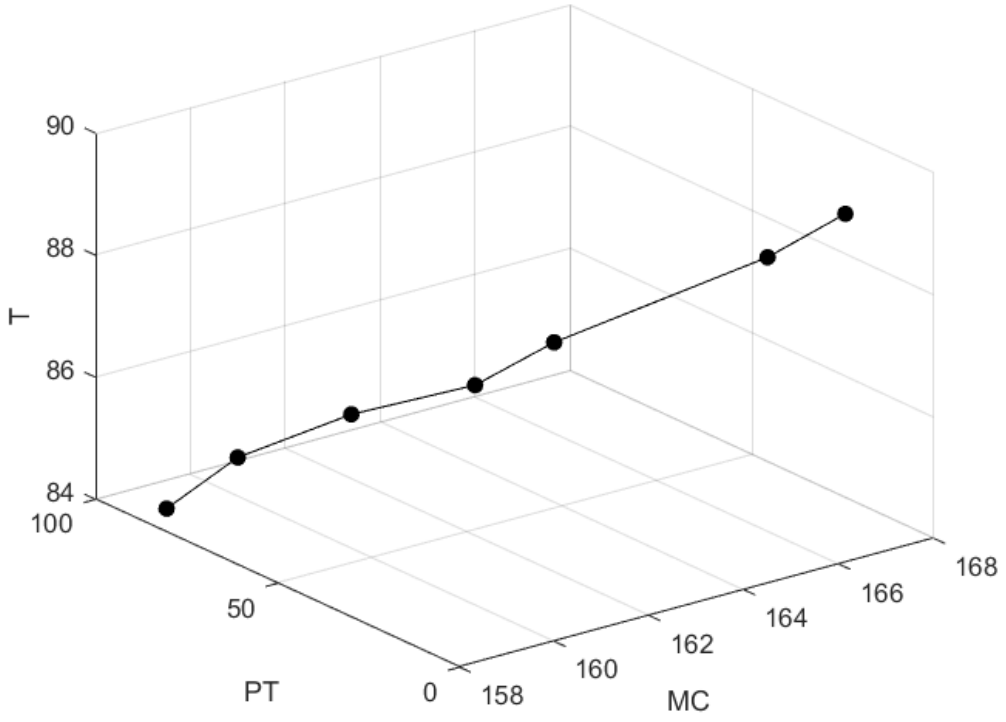


图 4.14 (m=100,n=200) STPF 不同阈值下最优解的 PT 值和场景均值

图中的 PT 轴即为对应的 $PT(\mathrm{X}^b,T)$，反应了当前解的鲁棒性，而 MC 轴的数据则为不同解 $MC(\mathrm{X}^b)$ 的值，反应了结果的优化性。从图中可以看出，阈值越低，求解 STPF 得到的 X^b 均值场景 $MC(\mathrm{X}^b)$ 的值也越小，系统整体的期望性能越强。这表明降低阈值时更多坏场景下的性能得到了抑制，整个系统实现了对更优期望性能的追求。但降低阈值会导致每个解对应的坏场景增加，相应的 $PT(\mathrm{X}^b,T)$ 也会增加，系统鲁棒性下降。反之亦然。这一结果也对两阶段模型 TSPTM 的性质进行了验证，充分说明了在一致并行机调度问题中该两阶段阈值坏场景集模型可以实现对系统鲁棒性和优化性的对立统一，通过求解不同的阈值下的 PT 鲁棒解可以实现风险偏向和最优性能中追求之间不同程度的折中，为决策者提供不同的决策方案。

4.5 本章小结

本章为两阶段阈值坏场景集模型设计了两阶段代理求解框架。为了求解该两阶段模型，本文根据不同阶段目标函数和求解精确的不同选用了两种算法组合起来的两阶段算法进行求解。首先在第一阶段提出了均值场景的概念，将场景均值模型 MCM 转换为均值场景下的确定性一致并行机调度问题后利用割平面法进行求解。在第二阶段采用第三章提出的基于合并坏场景邻域的布谷鸟算法 UNCSA 进行求解。

本章对两阶段的算法分别进行了测试计算。先对割平面法进行了大量的仿真，通过实验数据证明了该算法可以在合理的时间内对本文选用的不同规模下的一致并行机算例进行有效的求解。然后将求解结果用于第二阶段，通过将 UNCSA 算法与已有的三种智能优化算法进行对比说明了 UNCSA 算法在求解第二阶段问题时的有效性和优势，并通过不同算法的求解结果对比说明了减少第二阶段伪有效解的方式和意义。最后并利用仿真结果验证了整个两阶段模型的性质。证明了该两阶段阈值坏场景模型实现了系统鲁棒性和优化性的对立统一。

第五章　总结与展望

生产调度问题作为加工制造的核心问题，可以为企业的生产制造提供合理的生产计划，实现对生产资源和社会资源的合理分配，从而提高生产效率增加经济收益。因而对生产调度问题的研究有着重要的现实意义和研究价值。一致并行机调度问题作为生产调度中的典型问题类型，广泛地存在于各种实际的生产环境中，是联系不同调度问题的纽带之一。对一致并行机调度问题开展研究可以为其他的调度问题提供基础和参考。在实际的生产环境中充满了各种无法预知或避免的不确定因素，相比确定性生产调度问题，如何处理不确定生产参数并设计合适的不确定生产模型和求解算法更具现实意义。本文研究加工时间不确定的一致并行机鲁棒调度问题，利用离散场景来描述不确定的加工时间。本文在传统的最坏场景鲁棒优化模型基础上进行拓展，建立了一个两阶段阈值坏场景集惩罚模型。本文根据求解问题的特点，设计了一种面向问题的合并坏场景集邻域构造方式，并将采用该邻域构造方式的局部搜索与传统的布谷鸟算法结合，提出了一种合并坏场景集邻域的混合布谷鸟算法 UNCSA。本文主要做了以下工作：

1）本文用离散场景描述一致并行机鲁棒调度问题中的不确定加工时间，设计了一种用于一致并行机的两阶段阈值坏场景集惩罚模型。第一阶段进行阈值的合理取值区间计算，第二阶段则是在给定一个合理的阈值作为基准性能的情况下，筛选出场景集中性能不达标的坏场景，将坏场景下的性能与阈值的差值的平方作为该场景下的惩罚项。通过优化所有坏场景下惩罚值的总和来获得抵御坏场景性能下降风险的鲁棒解。最后将不同的阈值和对应的鲁棒解组成有效解集提供给决策者，可以让决策者根据鲁棒性和优良性能不同的偏好做出不同的抉择。

2）本文为该两阶段模型设计了一个代理两阶段求解框架。在第一阶段求解离散场景下的均值场景模型，将求得的最佳均值作为第二阶段阈值的起始值。第二阶段则是从起始值开始以一定的步长增加阈值，生成不同阈值下的最小坏场景惩罚值供决策者选择。为了求解该两阶段模型本文设计了一个两阶段算法，首先给出均值场景的概念，将离散场景下的一致并行机调度问题转换成单一均值场景下的确定性一致并行机问题然后采用割平面法进行求解。在第二阶段则先根据一致并行机调度问题的特点设计了两种邻域构造方式，然后在此基础上根据阈值坏

场景集模型的特点，设计了一种面向问题的合并坏场景集邻域构造方式，将所有坏场景下产生的邻域解合并到一个集合中进行邻域搜索。同时将该邻域搜索与布谷鸟算法相结合，在布谷鸟算法强大的全局搜索中加入局部搜索来提升对解空间的搜索效率。本文将该混合布谷鸟算法 UNCSA 与已有的其他三种智能优化算法进行了仿真对比，实验结果表明了 UNCSA 算法在和该合并坏场景集邻域构造方式在求解离散场景下一致并行机问题的阈值坏场景模型的有效性和优势。

制定合理的生产计划可以帮助企业减少不必要的消耗，增加生产效率提高企业竞争力。实际的生产环境中充满了不确定因素，建立合理的生产模型来处理各种不确定参数，并能根据决策者面对不确定参数时的不同决策偏好提供解决方案更加具有现实意义。本文主要研究了用离散场景描述不确定加工时间的鲁棒一致并行机调度问题，取得了一定的研究成果和进展，在本文研究问题的基础上还有很多可以进一步研究的地方，主要有以下几个方面：

1）本文研究了加工时间不确定的一致并行机调度问题。实际的生产过程中还存在其他的不确定生产参数，如交货期，工件到达时间，机器故障等，今后可以考虑更多的不确定参数对生产活动造成的影响并对其进行研究。

2）本文研究的一致并行机调度问题的目标函数是最大完工时间，实际的生产过程中存在着其他不同的生产目标，如工件延迟数量，总加工消耗，总滞后时间等。对单个或者多个目标函数进行研究，讨论不同目标的一致并行机调度问题的优化也是未来的研究方向之一。

3）本文设计的混合布谷鸟算法采用了在布谷鸟算法中加入局部搜索的方式来提升搜索效率。虽然通过兼顾全局搜索和邻域的方式增加了搜索的深度，扩大了，每代搜索的范围，相比单一的算法性有了很大的提升，但是依旧存在可以改进的地方。本文目前只采用了普通的邻域搜索方式，可以将布谷鸟算法与其他邻域算法进行结合，进一步提升求解效果。

4）本文主要研究了设计的模型在一致并行机调度中的应用，未来可以尝试在其他类型的调度问题背景下应用本文设计的鲁棒优化模型。

参考文献

[1] Rodammer F A, White K P. A recent survey of production scheduling[J]. IEEE transactions on systems, man, and cybernetics, 1988, 18(6): 841-851.

[2] Rocholl J, Mönch L. Decomposition heuristics for parallel-machine multiple orders per job scheduling problems with a common due date[J]. Journal of the Operational Research Society, 2019: 1-17.

[3] Lee C H. A new discrete electromagnetism-like mechanism algorithm for identical parallel machine scheduling problem with eligibility constraints in metal nuts manufacturing[J]. Arabian Journal for Science and Engineering, 2017, 42(8): 3609-3620.

[4] Güngör M, Ünal A T, Taşkın Z C. A parallel machine lot-sizing and scheduling problem with a secondary resource and cumulative demand[J]. International Journal of Production Research, 2018, 56(9): 3344-3357.

[5] Yamamoto M, Nof S Y. Scheduling/rescheduling in the manufacturing operating system environment[J]. International Journal of Production Research, 1985, 23(4): 705-722.

[6] Mignon D J, Honkomp S J, Reklaitis G V. A framework for investigating schedule robustness under uncertainty[J]. Computers & chemical engineering, 1995, 19: 615-620.

[7] McNaughton R. Scheduling with deadlines and loss functions[J]. Management Science, 1959, 6(1): 1-12.

[8] Dell'Amico M, Martello S. Optimal scheduling of tasks on identical parallel processors[J]. ORSA Journal on Computing, 1995, 7(2): 191-200.

[9] Shim S O, Kim Y D. A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property[J]. Computers & Operations Research, 2008, 35(3): 863-875.

[10] Haouari M, Jemmali M. Tight bounds for the identical parallel machine‐scheduling problem: Part II[J]. International Transactions in Operational Research, 2008, 15(1): 19-34.

[11] Mokotoff E. An exact algorithm for the identical parallel machine scheduling problem[J]. European Journal of Operational Research, 2004, 152(3): 758-769.

[12] Dell'Amico M, Iori M, Martello S, et al. Heuristic and exact algorithms for the identical

parallel machine scheduling problem[J]. INFORMS Journal on Computing, 2008, 20(3): 333-344.

[13] Chen Z L, Powell W B. Solving parallel machine scheduling problems by column generation[J]. INFORMS Journal on Computing, 1999, 11(1): 78-94.

[14] Garey M, Johnson D. Computers and intractability: A guide to the theory of NP-Completeness[M]. W. H. Freeman and Company 1979.

[15] Graham R L. Bounds on multiprocessing timing anomalies[J]. SIAM journal on Applied Mathematics, 1969, 17(2): 416-429.

[16] Lee W C, Wu C C, Chen P. A simulated annealing approach to makespan minimization on identical parallel machines[J]. The International Journal of Advanced Manufacturing Technology, 2006, 31(3-4): 328-334.

[17] Coffman, Jr E G, Garey M R, Johnson D S. An application of bin-packing to multiprocessor scheduling[J]. SIAM Journal on Computing, 1978, 7(1): 1-17.

[18] Lee C Y, Massey J D. Multiprocessor scheduling: combining LPT and MULTIFIT[J]. Discrete applied mathematics, 1988, 20(3): 233-242.

[19] Gupta J N D, Ruiz-Torres A J. A LISTFIT heuristic for minimizing makespan on identical parallel machines[J]. Production Planning & Control, 2001, 12(1): 28-36.

[20] Yamashita D S. Tabu search for scheduling on identical parallel machines to minimize mean tardiness [J]. Journal of intelligent manufacturing, 2000, 11(5):453-460.

[21] Alharkan I, Bamatraf K, Noman M A, et al. An order effect of neighborhood structures in variable neighborhood search algorithm for minimizing the makespan in an identical parallel machine scheduling[J]. Mathematical Problems in Engineering, 2018, 2018.

[22] Min L, Cheng W. A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines[J]. Artificial Intelligence in Engineering, 1999, 13(4): 399-403.

[23] Kashan A H, Karimi B. A discrete particle swarm optimization algorithm for scheduling parallel machines[J]. Computers & Industrial Engineering, 2009, 56(1): 216-223.

[24] Neto R F T, Godinho Filho M, Da Silva F M. An ant colony optimization approach for the parallel machine scheduling problem with outsourcing allowed[J]. Journal of Intelligent Manufacturing, 2015, 26(3): 527-538.

[25] Laha D, Gupta J N D. An improved cuckoo search algorithm for scheduling jobs on identical parallel machines[J]. Computers & Industrial Engineering, 2018, 126: 348-360.

[26] 顾幸生. 不确定性条件下的生产调度[J]. 华东理工大学学报,2000, 26(5), 441-446.

[27] Yamamoto M, Nof S Y. Scheduling/rescheduling in the manufacturing operating system environment[J]. International Journal of Production Research, 1985, 23(4): 705-722.

[28] Mignon D J, Honkomp S J, Reklaitis G V. A framework for investigating schedule robustness under uncertainty[J]. Computers & chemical engineering, 1995, 19: 615-620.

[29] Alimoradi S, Hematian M, Moslehi G. Robust scheduling of parallel machines considering total flow time[J]. Computers & Industrial Engineering, 2016, 93: 152-161.

[30] Ranjbar M, Davari M, Leus R. Two branch-and-bound algorithms for the robust parallel machine scheduling problem[J]. Computers & Operations Research, 2012, 39(7): 1652-1660.

[31] Cai X, Zhou S. Stochastic scheduling on parallel machines subject to random breakdowns to minimize expected costs for earliness and tardy jobs[J]. Operations Research, 1999, 47(3): 422-437.

[32] Zadeh L A. Fuzzy sets as a basis for a theory of possibility[J]. Fuzzy sets and systems, 1978, 1(1): 3-28.

[33] Balin S. Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation[J]. Information Sciences, 2011, 181(17): 3551-3569.

[34] Yeh W C, Lai P J, Lee W C, et al. Parallel-machine scheduling to minimize makespan with fuzzy processing times and learning effects[J]. Information Sciences, 2014, 269: 142-158.

[35] Behnamian J. Particle swarm optimization-based algorithm for fuzzy parallel machine scheduling[J]. The International Journal of Advanced Manufacturing Technology, 2014, 75(5-8):883-895.

[36] Anglani A, Grieco A, Guerriero E, et al. Robust scheduling of parallel machines with sequence-dependent set-up costs[J]. European journal of operational research, 2005, 161(3): 704-720.

[37] Allahverdi A, Aydilek H. Heuristics for the two-machine flowshop scheduling problem to minimize maximum lateness with bounded processing times[J]. Computers & Mathematics with Applications, 2010, 60(5):1374-1384.

[38] Kouvelis P, Yu G. Robust discrete optimization and its applications[M]. Spring, 1997.

[39] Li Z, Ierapetritou M G. Robust optimization for process scheduling under uncertainty[J]. Industrial & Engineering Chemistry Research, 2008, 47(12):4148-4157.

[40] Sarin S C, Nagarajan B, Jain S, et al. Analytic evaluation of the expectation and variance of different performance measures of a schedule on a single machine under processing time variability[J]. Journal of combinatorial optimization, 2009, 17(4):400-416.

[41] Liu M, Liu X, Zhang E, et al. Scenario-based heuristic to two-stage stochastic program for the parallel machine ScheLoc problem[J]. International Journal of Production Research, 2019, 57(6):1706-1723.

[42] Xu X, Cui W, Lin J, et al. Robust makespan minimisation in identical parallel machine scheduling problem with interval data[J]. International Journal of Production Research, 2013, 51(12):3532-3548.

[43] Drwal M, Rischke R. Complexity of interval minmax regret scheduling on parallel identical machines with total completion time criterion[J]. Operations Research Letters, 2016, 44(3): 354-358.

[44] Wang S, Cui W. Approximation algorithms for the min-max regret identical parallel machine scheduling problem with outsourcing and uncertain processing time[J]. International Journal of Production Research, 2020:1-14.

[45] Wang B, Wang X, Lan F, et al. A hybrid local-search algorithm for robust job-shop scheduling under scenarios[J]. Applied Soft Computing, 2018, 62:259-271.

[46] Wang S, Zou H, Wang S. A Tabu-GA-based parallel machine scheduling with restrained tool resources[J]. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2020:0954405420928691.

[47] Lee W C , Wang J Y , Lee L Y . A hybrid genetic algorithm for an identical parallel-machine problem with maintenance activity[J]. Journal of the Operational Research Society, 2015.

[48] Chen J, Pan Q K, Wang L, et al. A hybrid dynamic harmony search algorithm for identical parallel machines scheduling [J]. Engineering Optimization, 2012, 44(2):209-224.

[49] 何嘉. 异速并行机系统生产调度与预防性维护集成研究[D]. 重庆大学, 2016.

[50] Cheng T C E, Sin C C S. A state-of-the-art review of parallel-machine scheduling research[J].

European Journal of Operational Research, 1990, 47(3):271-292.

[51] Edis E B, Oguz C, Ozkarahan I. Parallel machine scheduling with additional resources: Notation, classification, models and solution methods[J]. European Journal of Operational Research, 2013, 230(3):449-463.

[52] Mulvey J M, Vanderbei R J, Zenios S A. Robust optimization of large-scale systems[J]. Operations research, 1995, 43(2):264-281.

[53] Wang S, Cui W. Approximation algorithms for the min-max regret identical parallel machine scheduling problem with outsourcing and uncertain processing time[J]. International Journal of Production Research, 2020: 1-14.

[54] Drwal M, Roman, Complexity of interval minmax regret scheduling on parallel identical machines with total completion time criterion[J]. Operations Research Letters, 2016.

[55] Yang X S, Deb S. Cuckoo search via Lévy flights[A].2009 World congress on nature & biologically inspired computing (NaBIC)[C]. Coimbatore, India: IEEE, 2009:210-214.

[56] Abdel-Basset M, Hessin A N, Abdel-Fatah L. A comprehensive study of cuckoo-inspired algorithms[J]. Neural Computing and Applications, 2018, 29(2):345-361.

[57] Valian E, Valian E. A cuckoo search algorithm by Lévy flights for solving reliability redundancy allocation problems[J]. Engineering Optimization, 2013, 45(11):1273-1286.

[58] Wang H, Wang W, Sun H, et al. A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems[J]. Soft Computing, 2017, 21(15):4297-4307.

[59] Guo P, Cheng W, Wang Y. Parallel machine scheduling with step-deteriorating jobs and setup times by a hybrid discrete cuckoo search algorithm[J]. Engineering Optimization, 2015, 47(11):1564-1585.

[60] Shehab M, Khader A T, Laouchedi M. A hybrid method based on cuckoo search algorithm for global optimization problems[J]. Journal of Information and Communication Technology, 2018, 17(3):469-491.

[61] Sheng Z, Wang J, Zhou S, et al. Parameter estimation for chaotic systems using a hybrid adaptive cuckoo search with simulated annealing algorithm[J]. Chaos: An Interdisciplinary Journal of Nonlinear Science, 2014, 24(1):013133.

[62] Payne R B, Sorensen M D. The cuckoos[M]. Oxford University Press, 2005.

[63] Reynolds A M. Cooperative random Lévy flight searches and the flight patterns of honeybees[J]. Physics letters A, 2006, 354(5-6):384-388.

[64] Schreier A L, Grove M. Ranging patterns of hamadryas baboons: random walk analyses[J]. Animal Behaviour, 2010, 80(1):75-87.

[65] Ramos-Fernández G, Mateos J L, Miramontes O, et al. Lévy walk patterns in the foraging movements of spider monkeys (Ateles geoffroyi)[J]. Behavioral ecology and Sociobiology, 2004, 55(3): 223-230.

[66] Austin D, Bowen W D, McMillan J I. Intraspecific variation in movement patterns: modeling individual behaviour in a large marine predator[J]. Oikos, 2004, 105(1):15-30.

[67] Atkinson R P D, Rhodes C J, Macdonald D W, et al. Scale‐free dynamics in the movement patterns of jackals[J]. Oikos, 2002, 98(1):134-140.

[68] Yang X S , Deb S . Engineering Optimisation by Cuckoo Search[J]. International Journal of Mathematical Modelling & Numerical Optimisation, 2010, 1(4):330-343.

[69] Yang X S , Deb S . Cuckoo Search: Recent Advances and Applications[J]. Neural Computing & Applications, 2014, 24(1):169-174

[70] Yang X S. Nature-inspired metaheuristic algorithms[M]. Luniver press, 2010.

[71] Mantegna R N. Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes[J]. Physical Review E, 1994, 49(5):4677.

[72] 叶春明, 李永林, 刘长平. 新型仿生群智能算法及其生产调度应用[M]. 科学出版社, 2015.

[73] 陈怡萍. 布谷鸟算法及应用研究[D]. 浙江大学, 2019.

[74] Daniels R L, Carrillo J E. β-Robust scheduling for single-machine systems with uncertain processing times[J]. IIE transactions, 1997, 29(11): 977-985.

[75] Lim H T, Ramli R. Recent advancements of nurse scheduling models and a potential path[J]. 2010.

[76] Liang J J , Pan Q K , Tiejun C , et al. Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer[J]. The International Journal of Advanced Manufacturing Technology, 2010.

[77] Fatih Tasgetiren M, Liang Y C, Sevkli M, et al. Particle swarm optimization and differential

evolution for the single machine total weighted tardiness problem[J]. International Journal of Production Research, 2006, 44(22):4737-4754.

[78] Tang L , Wang X . An Improved Particle Swarm Optimization Algorithm for the Hybrid Flowshop Scheduling to Minimize Total Weighted Completion Time in Process Industry[J]. IEEE Transactions on Control Systems Technology, 2010, 18(6):1303-1314.

[79] Rui Z . An Artificial Bee Colony Algorithm Based on Problem Data Properties for Scheduling Job Shops[J]. Procedia Engineering, 2011, 23(1):131-136.

[80] Kouvelis P, Daniels R L, Vairaktarakis G. Robust scheduling of a two-machine flow shop with uncertain processing times[J]. IIE Transactions, 2000, 32(5):421-432.

[81] Wang B, Xia X, Meng H, et al. Bad-scenario-set robust optimization framework with two objectives for uncertain scheduling systems[J]. IEEE/CAA Journal of Automatica Sinica, 2017, 4(1):143-153.

作者在攻读硕士期间公开发表的论文

[1] Wang B, Xie H, Xia X, et al. A NSGA-II algorithm hybridizing local simulated-annealing operators for a bi-criteria robust job-shop scheduling problem under scenarios[J]. IEEE Transactions on Fuzzy Systems, 2018, 27(5):1075-1084.

[2] Wang B, Wang X, Xie H. Bad-scenario-set robust scheduling for a job shop to hedge against processing time uncertainty[J]. International Journal of Production Research, 2019, 57(10): 3168-3185.

[3] Xie H, Wang B. An exact algorithm for identical parallel machine scheduling under scenarios[C]. 2020 3rd World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM). 2020:341-344

致　谢

时间总在不经意间匆匆流逝。窗外春光明媚，春意渐浓，又是一年毕业季，我的研究生生学习生涯也即将画上句号。回首自己这几年的求学经历，有过挣扎和迷茫，也走过弯路留下遗憾，但更多的是成长和蜕变。在这里，我想对陪伴我成长，给予我帮助的人表示由衷的感谢和敬意。

首先我要感谢我的导师王冰教授。她在研究生期间一直悉心指导我，从学术和思想两方面引导我成为一名合格的研究生。在研一刚进来时，王老师就为我规划好了研究课题和方向，每周她都会通过周报的形式来了解我们的学习进度和动态，并以此做出相应的学习计划调整。王老师会定期召开组会，在组会中毫无保留地分享她自己多年的学术成果。我曾经有一段时间心态非常浮躁，沉迷网络游戏逃避现实，耽误了自己的课题研究荒废了自己的学术能力，以至于自己的整个研究陷入迟滞无法推进。王老师及时发现了这种状况，多次与我单独交谈，给予我悉心的指导和关怀帮助我回到正轨。我研究生论文的背后，倾注了王老师大量的心血。没有她的悉心教导，我也无法顺利地完成我的学业。而王老师严谨求实的工作作风和勤奋自强的生活态度也必将在往后的日子里一直影响指引着我。

其次我要感谢研究生期间帮助和陪伴我的同学们。感谢师姐叶俏妮，师兄刘利甲、邬波，他们从我进入课题组开始就一直给予我引导和帮助。感谢同门贺玉凤，师弟倪政斌、苏润、杨世秦、胡恒督、冯凯，师妹陈晔琳、吕斐然，大家在学习上的互相帮助共同成长让我难以忘怀。感谢曾经的室友柴华、夏鑫、刘兵，他们在生活上对我的关心和包容，祝福他们生活顺利，前程似锦。

我还要感谢一直陪伴我的朋友们，在我消沉沮丧的时候他们一直不离不弃给我支持和鼓励。感谢洪流、丁帆、张超、吕祖鹏、林涛、焦一珽、庞煜晨、黄旭东，感谢李博、易慕白和张思远，还有小鱼，感谢西川的七人组。你们一直的陪伴给了我生活的意义。

最后要感谢我的家人们，他们是我成长路上坚实的后盾，他们的鼓励和支持是我一直前进的动力。

研究生生活的结束意味着人生一个阶段的完成，也意味着下一个阶段的启程。在春暖花开的日子我们走出校门奔向社会，祝我自己的未来明媚美好。