华中科技大学

研究生（选题）报告

题 目：基于提前和延误惩罚的单机调度问题启发式算法研究

学　　号　M201572797

姓　　名　覃　涛

专　　业　计算机软件与理论

指 导 教 师　吕志鹏

院（系、所）　计算机科学与技术学院

填 表 注 意 事 项

一、本表适用于攻读硕士学位研究生选题报告、学术报告，攻读专业硕士学位研究生实践环节报告，攻读博士学位研究生文献综述、选题报告、论文中期进展报告、学术报告等。

二、以上各报告内容及要求由相关院（系、所）做具体要求。

三、以上各报告均须存入研究生个人学籍档案。

四、本表填写要求文句通顺、内容明确、字迹工整。

一、课题来源、目的和意义

从科技的角度来看，未来二、三十年人类社会将演变成一个智能社会，其深度和广度我们还想象不到[1]。近些年来，得益于大数据技术的发展，人工智能逐步兴起，并开始影响人们的生活，智能家居、无人驾驶汽车、个性化推荐、人脸识别等等，都是人工智能给人类带来的科技福利。在大数据和人工智能时代的大背景下，信息科学、机器学习和运筹学分别扮演者不同的角色。数据的采集、挖掘、存储和管理属于信息科学的范畴；拿到数据后，根据数据做分析得出某些规律，这属于统计学习、机器学习甚至深度学习的范畴；而根据数据总结出的规律，决策人需要做决定的时候，既需要考虑方案的合理性，也需要考虑约束条件的限制，需要考虑成本代价或者最终受益，希望在复杂的规律中找到最优化的决策，这就是运筹学要解决的问题。可以预见的是，随着人们对更深层次更全局性优化决策的需求，运筹学必然会扮演着更重要的角色。

在有限数量可行解的集合中找出最优解的一类问题称为组合优化问题，它是运筹学的一个重要分支。所研究的问题涉及信息技术、经济管理、工业工程、交通运输、通讯网络等诸多领域。组合优化问题求解的是离散型变量的优化问题，其一般形式可以描述为：在满足一定的约束或者限制条件下，求取使得给定的目标函数值最大或者最小的解。

单机调度问题（single machine scheduling problem）是组合优化问题的一种，具有较高的理论研究价值，而且实际应用中有很多相关问题亟待解决，因此吸引了全世界研究人员的广泛关注。单机调度问题主要研究的是如何在一台给定的机器上安排一组加工任务，并希望该调度方案使得绩效度量达到最优。这类问题在实际生产生活中有广泛的应用，如适时生产理念（just-in-time，简称 JIT）和供应链管理[2]。适时生产理念提倡物品按需按量按时生产，无论是提前或者延后生产都不建议，都会对生产效益产生不好的影响，太早工厂库存压力大，太晚影响订

[1] 华为任正非 2016 年在全国科技创新大会上的汇报开篇发言

单进度，因此理想的生产计划是所有的工作都在规定的时刻完成。单机调度问题在供应链管理中也有应用，将供应商比做生产机器，每道作业或者工序就是客户的配送任务，在实际中，强调供应商和客户之间的协调，要求尽量在指定的时间完成客户的配送，太早客户库存容量限制不能接受，太晚影响供应商信誉，也可能会使客户的其它安排滞后，所以安排合理的配送方案使得供应需求按时满足在供应链管理中就显得尤为重要。另外，在 CPU 调度、车间作业调度和机场飞机起降调度等问题中，单机调度问题都有广泛的应用，因此解决这一问题可以带来巨大的经济效益。已经证明，单机调度问题属于典型的 NP 难问题[3]。求解这一问题具有重要的理论研究价值。

二、国内外研究现状及发展趋势

单机调度问题是比较经典的组合优化问题，业已吸引了国内外众多学者的研究。生产调度问题的研究历史较为悠久，最早是由 Johnson 于 20 世纪 50 年代提出，距今 60 多年的历史。根据实际生产生活中的需要，后来诸多学者提出了几种不同求解目标的单机调度问题，并总结出了科学描述生产调度问题的方法。1967 年 Conway 提出了调度问题的四参数表示法，后来 Graham、Lawler 等人提出了更好的三参数表示法（$\alpha|\beta|\gamma$），其中α用来描述机器的数量、类型等机器环境信息，β用于描述工件的加工约束信息，γ用于描述目标函数。

单机调度问题是已被证明的 NP 难问题，由于其复杂性和多样性，很难找到一种通用的求解算法。一般地，解决单机调度问题的算法主要分为两类：第一种是精确算法，利用分支定界或者动态规划来求解；第二种是启发式算法，通过解的迭代更新，不断改进解。

- 精确算法

一般来说，求解单机调度问题的精确算法主要是分支定界（Branch and Bound）和动态规划（Dynamic Programming）两种。1990 年 Abdul-Razaq 提出了两种动态规划方法和四中分支定界算法来求解考虑带权延误惩罚的单机调度问题。Potts 和 Van Wassenhove 提出了求解延误惩罚权值相同的单机调度问题的动态规划算

法，亦提出了延误惩罚权值不同时的分支定界算法。Sourd 提出了一种分支定界法来求解具有相同预期完成时间的单机调度问题，并同时考虑了提前时间惩罚和延误惩罚。对于考虑二次延误惩罚的单机调度问题，Valente 基于工序完成时间的松弛提出了一种求解单机调度问题下界的方法，并采用合适的下界，结合插入探测，提出了求解该问题的分支定界算法。Schaller 针对该类问题提出了几种分支定界算法，这些算法通过计算紧凑的下界和测试两种优势条件来组合完成。最近，由 Tanaka 和 Araki 提出的一种精确算法找到了由 Cicirello 生成的考虑带加工准备时间和加权延误惩罚的单机调度问题的全部 120 算例的最优解；并用相同的算法求解延误惩罚不加权的单机调度问题，由 62/64 个算例得到了最优解。

动态规划或者分支定界等精确算法求解时一般可以获得较好的优度，但往往计算时间较长，对于较大规模的算例，需要几天甚至更久的时间，因此在实际中很难得到应用。

- 启发式算法

启发式算法主要包括局部搜索、禁忌搜索、遗传算法和混合算法等，在很多类型的单机调度问题上都有良好的计算效果。Potts 和 Van Wassenhove 提出了多种启发式算法来求解不带准备时间的单机调度问题，并发现通过成对交换工件加工位置是不错的策略。Holsenback 提出了一种求解加权延误惩罚的单机调度问题，处理的问题规模可以达到 50 个工件数量。Feo 提出了一种贪婪自适应搜索算法（GRASP）来求解带准备成本和线性延误惩罚的单机调度问题。Valente 和 Gonçalves 提出了基于随机关键字的几种遗传算法，这些遗传算法大致相同，只是在生成初始解和利用局部搜索收敛时有差异。Valente and Schaller 针对考虑机器准备时间和不考虑准备时间的单机调度问题，均提出了几种求解的启发式算法[20]。Rubin 和 Ragatz 提出了一种遗传算法（GA）用来求解最小化总延迟的单机调度问题。Tan 和 Narasimhan 提出一种模拟退火算法（SA）来解决带准备时间的单机调度问题，Armentano 和 Mazzini 同样则提出用遗传算法（GA）解决该问题，Franca 等人则提出了一种混合进化算法（Memetic）算法解决考虑准备时间的单机调度问题。

对于既考虑准备时间也考虑延误惩罚权重的单机调度问题，也有众多的学者

参与研究。Cicirello 和 Smith 生成了 120 个带准备时间带延误惩罚权重的单机调度问题算例，每个算例包括 60 个工件；并且他们还分析了一些随机抽样方法与模拟退火算法搭配在一起的性能。Lin 和 Ying 对比了遗传算法（GA）、禁忌搜索算法（TS）以及模拟退火算法（SA）在求解该类单机调度问题时的表现。Valente 提出了几种基于束搜索的启发式算法求解带二次延误惩罚的单机调度问题，这些启发式算法既包括利用优先级和总惩罚值来进行评估的束搜索算法，也有带可变束和过滤宽度的变体。近年来 Tasgetiren 等人提出的离散差分进化算法（Discrete Differential Evolution，DDE）以及 Kirlik 和 Oguz[28]提出的变邻域搜索算法（GVNS）大幅度的改进了该类单机调度问题算例的解。2014 年，Xu 提出了基于“块移动”迭代局部搜索算法和混合进化算法来求解，在很多该类单机调度问题的标准算例上得到了当前最好解。

三、关键理论和技术

在实际生产生活中，往往会遇到一些大规模的复杂优化问题，如机场航班调度、停机位分配问题、排课表问题、车间作业调度、圆形 packing 问题和负载均问题等等，这些问题都是 NP 难问题。目前求解 NP 难问题的主要算法有精确算法、近似算法和启发式算法三种，应用最有效最广泛的当属启发式算法。启发式算法（Heuristic）指的是人们受大自然的启发，通过总结经验和规则来求解问题的算法。实际中的组合优化问题往往求解难度很大，而且在大多数场景下，要求能快速决策，给出解决方案。一般地，精确算法很难在比较短的时间内找到较好的解，而启发式算法通过对问题的分析以及结合实际中的方案规则，能在合理的时间内得到较好的解，但启发式算法的求解性能缺乏理论支持，往往是通过实际求解结果来验证其性能。

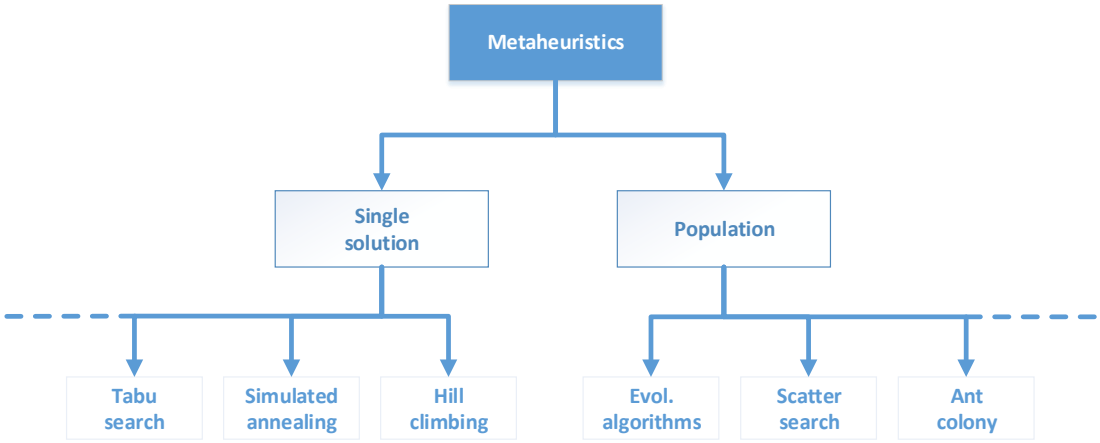

图 1 启发式算法族谱

图 1 给出了元启发式算法族谱，启发式算法一般分为个体优化算法和种群优化算法。个体优化算法包括局部搜索（又称为爬山法）、禁忌搜索、模拟退火等等，种群优化算法包括遗传算法、蚁群算法、散射搜索算法等等。

图 2 给出了启发式算法的设计空间示意图。对启发式算法来说，好的算法设计既要考虑到解的集中性，也要考虑到解的疏散性。所谓集中性，指的是解能够快速迭代更新优化，更快找到或者收敛到局部最优解；所谓疏散性，指的是搜索更广阔的解空间，更有利于增加搜索到全局最优解或者更好解结构的可能性。随机搜索算法中，疏散性的体现最强，搜索的空间会很大，但解的质量变化也很大，不会收敛；局部搜索中集中性的体现最强，随着搜索的进行，解会不断下降优化。一般来说个体优化算法更着重于体现解的集中性，但并非完全不考虑解的疏散性，一般会通过扰动或者禁忌表来增强解的疏散性。种群优化算法也并非完全指考虑解的疏散性，它通过解的交叉来扩展解空间，但依然会通过一些简单快速的局部搜索算法来使种群中的个体收敛到更优的解。

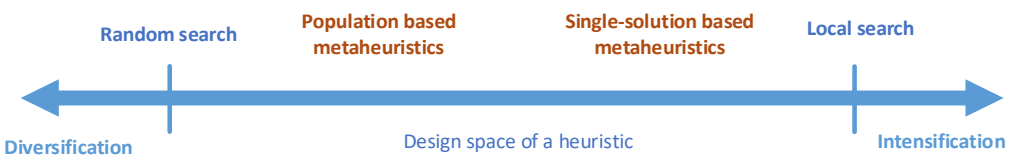

图 2 启发式算法的设计空间

拟采用带禁忌扰动的迭代局部搜索来求解基于提前惩罚和延误惩罚的单机调度问题，下面介绍下局部搜索和禁忌搜索的概念。

- **局部搜索**

局部搜索，顾名思义，就是通过有效的搜索策略，找到局部最优解。局部搜索的主要过程是从一个初始解出发，通过在解空间中搜索当前解的邻域，找到邻域中更好或者最好解，并用找到的新解替换当前解，重复上述过程直到解不再更新为止。

局部搜索算法设计主要集中在三个方面：邻域结构设计，评估函数设计以及达到局部最优后如何进一步拓展解空间。所谓邻域结构，指的是从当前解出发，通过简单的邻域动作对当前解进行幅度较小的改动，得到邻域解空间。图 3 给出了局部搜索示意图，最左表示当前解，右边表示邻域候选解，局部搜索就是不断从当前解出发构造邻域得到候选解集，并从中选择某个解作为新的当前解，重复这一过程就是局部搜索的主要流程。一般来说，邻域解是通过在当前解的基础上，对解的构成做比较小的改动（称为邻域动作），得到的具备新结构的解。当然，做邻域动作需要保证解的合法性不受影响。邻域空间越小，从邻域中找到最优解的速度越快，但搜索的范围小，不一定能覆盖到好的解，很容易陷入局部最优；邻域空间越大，从邻域中找到最优解的速度越慢，但搜索的解数量多，更容易持续更新当前解。因此，好的邻域结构设计也要考虑集中性和疏散性的平衡，既需要保证搜索的效率，也要尽可能搜索到更多的优质解，避免很快陷入局部最优。评估函数指的在确定邻域结构之后，如何从邻域中选择新的解替换当前解，这依赖于评估函数的选择。评估函数需要从领域解结构中挑选出优质的解，一般来说评估函数直接反应了目标函数的值或者目标函数的变化量，旨在不断得到更优的解。评估函数的确立也要注意搜索的效率，由于局部搜索中往往需要大量重复使用评估函数来评估，高效的评估函数可以极大的提高搜索效率。

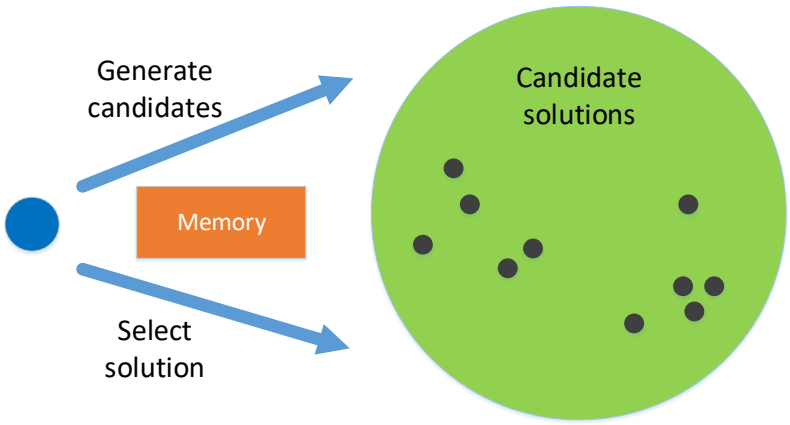

图 3 局部搜索示意图

局部搜索一个重要的特征就是算法在执行过程中会逐步陷入局部最优，即当前解就是邻域中的最优解时，算法将无法搜索更好的解，因此需要设计某种机制，如对解进行扰动（较大程度改变解的结构）扩展新的搜索空间。扰动机制的作用是希望跳出局部最优，因此扰动的幅度不能太小，避免又回到局部最优解；扰动的幅度也不宜过大，幅度过大的扰动类似于重新生成初始解，之前搜索得到的解毫无作用，不易于保持优良的解结构，会因为无效率的搜索大幅度降低算法的计算速度和性能，也不利于进一步改进解。

- **禁忌搜索**

禁忌搜索（Tabu Search 或 Taboo Search，简称 TS）的思想最早由 Glover(1986)提出[36]，它是对局部领域搜索的一种扩展。禁忌搜索的主要思想是通过特殊的存储结构记录搜索过的解空间。当然，一般来说很难直接将搜索过的解完全保存下来，通常是通过记录一些邻域动作，来判断当前的邻域动作是否被禁忌。

禁忌搜索的大致流程和局部搜索很相似，也是从一个初始解出发，通过搜寻邻域中的解来不断更新当前解。但与之不同的是，禁忌搜索通过禁忌表来记录搜索区域避免迂回搜索。在禁忌搜索的过程中，会将搜索过的解或者动作记录在禁忌表中，禁忌表中的元素也并非一直被禁忌，每个元素的禁忌时间称为禁忌长度，当迭代一段时间之后允许禁忌表中的元素被解禁，避免漏掉结构性良好的解。由于禁忌表的存在，禁忌搜索在每次扩展邻域的时候会将邻域解集合划分为受禁忌

的候选集和不受禁忌的候选集，确立邻域最优解会从不受禁忌的候选集中的元素中挑选，但也存在一些例外情况，称为特赦准则。所谓特赦准则，指的是有一些邻域解虽然被禁忌，但为了获得更好的解，也必须选择，所以一般认为如果当前解的目标函数值优于找到的最好解的目标函数值，即便该解被禁忌，也允许选择该动作来替换当前解。

从禁忌搜索的流程上来看，禁忌表的设计显得尤为重要。一方面，禁忌表要对搜索过的解空间进行封闭，因此禁忌表既不能漏掉太多搜索过的空间，不然难以避免迂回搜索；另一方面亦不能将未搜索过的解标记为禁忌状态，否则可能会因为错误的标记漏掉结构较好的解。禁忌表不但需要保存禁忌元素，还要判断邻域解是否处于禁忌状态，因此保证禁忌判断的效率也显得尤为关键。由于组合优化问题的结构一般比较复杂，即便是本文研究的单机调度问题，其解也是工件的序列，但如果禁忌表中直接保存单机调度问题的解，那可能搜索的绝大部分时间开销都是在判断当前工件序列是否和解中的一致，显然算法的性能大打折扣，因此通常来说会将一些邻域动作加入到禁忌表，既能通过判断邻域动作是否在禁忌表中来快速判断当前解是否被禁忌，也能有效地避免迂回搜素。

研　究　生　签　字 _____

指　导　教　师　签　字 _____

院(系、所)领导签字 _____

年　　月　　日