带有资源冲突的 *Seru* 在线并行调度算法

江煜舟 [1] 李冬妮 [1] 靳洪博 [1] 殷 勇 [2]

摘 要 随着大规模定制的市场需求日趋显著, 赛如生产系统 (*Seru* production system, SPS) 应运而生, 逐渐成为研究和应用领域的热点. 本文针对带有资源冲突的 *Seru* 在线并行调度问题进行研究, 即需要在有限的空间位置上安排随动态需求而构建的若干 *Seru*, 以总加权完工时间最小为目标, 决策 *Seru* 的构建顺序及时间. 先基于平均延迟最短加权处理时间 (Average delayed shortest weighted processing time, AD-SWPT) 算法, 针对其竞争比不为常数的局限性, 引入调节参数, 得到竞争比为常数的无资源冲突的 *Seru* 在线并行调度算法. 接下来, 引入冲突处理机制, 得到有资源冲突的 *Seru* 在线并行调度算法, αAD-I (α-average delayed shortest weighted processing time-improved) 算法, 特殊实例下可通过实例归约的方法证明其竞争比与无资源冲突的情况相同. 最后, 通过实验, 验证了在波动的市场环境下算法对于特殊实例与一般实例的优越性.

关键词 赛如生产系统, 在线调度, 竞争比, 实例归约, 总加权完工时间

引用格式 江煜舟, 李冬妮, 靳洪博, 殷勇. 带有资源冲突的 *Seru* 在线并行调度算法. 自动化学报, 2022, **48**(2): 444−459

DOI 10.16383/j.aas.c190698

An Online Algorithm for Parallel Scheduling of *Serus* With Resource Conflicts

JIANG Yu-Zhou[1] LI Dong-Ni[1] JIN Hong-Bo[1] YIN Yong[2]

Abstract With the remarkably increase of mass customization, there comes the *seru* production system (SPS), which has become a hotspot in both the research and the application fields. This paper discusses the online parallel scheduling problem of *serus* with resource conflicts, which aims at scheduling *serus* that are generated with dynamic demands on limited space to minimize the total weighted completion time. First, we consider online parallel scheduling of *serus* without resource conflicts. Based on the average delayed shortest weighted processing time (AD-SWPT) algorithm, an adjustment parameter is introduced and an optimization algorithm with a constant competitive ratio is proposed. Then for online parallel scheduling of *serus* with resource conflicts, an α-average delayed shortest weighted processing time-improved (αAD-I) algorithm is proposed, whose competitive ratio is proved to be the same as the one without resource conflicts in special cases via instance reduction. Computational experiments are implemented to test and verify the superiority of our algorithm under both special instances and general instances.

Key words *Seru* production system, online scheduling, competitive ratio, instance reduction, total weighted completion time

Citation Jiang Yu-Zhou, Li Dong-Ni, Jin Hong-Bo, Yin Yong. An online algorithm for parallel scheduling of *serus* with resource conflicts. *Acta Automatica Sinica*, 2022, **48**(2): 444−459

随着大规模定制发展的趋势, 传统的生产系统, 如流水线 (Flow line)、丰田生产系统 (Toyota production system, TPS)、作业车间 (Job shop)、单元制造系统 (Cellular manufacturing system, CMS) 等, 难以适应对动态不确定市场的快速响应需求, 赛如生产系统 (*Seru* production system, SPS) 应运而生[1−3].

Yin 等[4] 的研究展示了传统生产系统转化为 SPS 的重要性, Liu 等[5−6] 的研究也表明 SPS 具有传统生产系统难以企及的先进性和发展前景[7]. 自二十世纪九十年代起, SPS 已经逐渐被亚洲的众多电子企业采用, 如三星、佳能、LG、索尼、松下、富士通、NEC、富士康等[8−10].

Seru 代指 SPS 下的最小生产单元, 脱胎自基于精益 (Lean) 思想[11] 的装配流水线, 一个 *Seru* 通常是生产一种或多种产品的装配单元, 包含若干设备和工人.

收稿日期 2019-10-09 录用日期 2020-02-07
Manuscript received October 9, 2019; accepted February 7, 2020
内蒙古自治区重大基础研究开放课题 (GZ2018KF001), 国家自然科学基金 (61763046) 资助
Supported by State Key Laboratory of Smart Manufacturing for Special Vehicles and Transmission Systems (GZ2018KF001), National Natural Science Foundation of China (61763046)
本文责任编委 贺威
Recommended by Associate Editor HE Wei
1. 北京理工大学计算机学院智能信息技术北京市重点实验室 北京 100081 中国 2. 同志社大学商学院 京都 602-0023 日本
1. Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China 2. Graduate School of Business, Doshisha University, Kyoto 602-0023, Japan

一个 SPS 至少包含一个 *Seru* . SPS 中的每一个 *Seru* 都能够频繁地在短时间内被重构, 这给 SPS 带来了极大的灵活性. 可以快速频繁地建立、改变、拆除和转化, 以响应频繁波动的市场需求[9−10].

SPS 运作管理的基本原则为面向 “组织”的准时生产原则 (Just-in-time organisation system, JIT-OS), 是 TPS 传统的面向 “物料”的准时生产原则 (Just-in-time material system, JIT-MS) 的延伸. JIT-MS 指在合适的时间地点投入合适的物料, 强调的是物料. 而 JIT-OS 强调的是组织, 对应到 SPS, 即在合适的时间地点构建合适的 *Seru* . 这让 SPS 可以通过调整生产组织结构快速获得相应的生产能力, 为重构的实施提供了有效的载体和途径[1].

SPS 的运作可以被划分为 *Seru* 构建与 *Seru* 调度两个部分, *Seru* 构建指如何依据订单任务对人员进行分配与组合, *Seru* 调度指如何在有限的空间下安排各个 *Seru* 的构建顺序及时间, 目前相关研究大都侧重于 *Seru* 构建. 如 Liu 等提出的解决工人分配问题的三段式启发模型[12]、Yu 等提出的以产品流通时间和总劳动时间为目标的一种非支配排序遗传算法[13]、Yu 等结合局部搜索算法提出的第二代非支配排序遗传算法[14]、吴旭辉等联合 *Seru* 构建与订单分配提出的一种协同进化算法[7]、贾凌云等[15]与田云娜等[16] 对跨单元调度问题的研究等.

目前对 *Seru* 调度这一方面的研究相对较少, 难以充分体现 SPS 调整结构的动态性, 但要想充分发挥出 SPS 的灵活性, 快速响应 “小批量, 多品种”市场的动态变化, 在提高 *Seru* 构建效率之外, 还需要考虑结构上的变化, 即 *Seru* 调度. 如何在有限的位置上安排 *Seru* 的构建顺序及时间也是 SPS 运作管理基本原则 JIT-OS 的一项重要内容.

据此, 本文对 *Seru* 在线并行调度问题展开了研究, 该问题具体是指, 将随时间动态构建的 n 个 *Seru* 安排到有限的 m 个位置上, 以总加权完工时间最小为目标, 在线决策各 *Seru* 的构建顺序及时间. 同时, 考虑到具体的生产环境, 为了增强算法的实用性, 本文还将对带有资源冲突的 *Seru* 在线并行调度问题进行讨论.

本文接下来的内容安排如下: 第 1 节, 给出具体的问题模型; 第 2 节, 给出 AD-SWPT 算法优化后得到的具有良好常数竞争比的在线算法; 第 3 节, 给出带有资源冲突的 *Seru* 在线并行调度算法, 并计算特殊实例下算法的竞争比; 第 4 节, 设计相关实验, 展示实验结果, 分析实验数据; 第 5 节为结论部分.

1　问题描述与分析

本文所研究的带有资源冲突的 *Seru* 在线并行调度问题是指, 如何在有限个 *Seru* 的可用位置上在线安排多个带有资源冲突的 *Seru* , 以提高生产效率, 具体描述如下:

有一系列的订单任务随着时间发布, 每一个订单任务对应一个 *Seru* , *Seru* 间可能存在资源冲突, 在无优先级的情况下, 将它们安排到 m 个 *Seru* 的可用位置上进行处理, 存在资源冲突的两个 *Seru* 不能同时被处理, 每一个 *Seru* 都对应一个订单任务发布时间 r_j, 一个处理时间 p_j 和一个权重 w_j, 所有关于 *Seru* 的信息在订单任务发布前都是未知的. 同样, *Seru* 的总数量也无法提前得知.

目标量是总加权完工时间 $\sum w_jC_j$, 其中 C_j 对应 *Seru* 的完工时间, 本文的目的是探寻一个使得总加权完工时间 $\sum w_jC_j$ 最小的在线调度算法.

在线算法的优劣通常用它的竞争比 ρ 来评价 (对任意实例, 算法的目标函数值都不大于 ρ 倍的最优离线目标值).

问题的基本假设如下:

1) 车间中有 m 个 *Seru* 的可用位置, 共有 n 个订单任务会发布 (m 与 n 均为正整数);

2) 所有发布的订单任务都可以被完成, 不考虑由工人技术或车间规模限制而导致的拒单;

3) 一个订单任务只能在一个 *Seru* 中完成, 不考虑拆分;

4) 一个 *Seru* 的可用位置同一时间段最多安排一个 *Seru* ;

5) 一旦 *Seru* 构建完成, 被安排到某一空的可用位置上, 在结束任务前 *Seru* 的位置不会发生改变, 任务进程不会被打断;

6) 不考虑由工具损坏或自然灾害等不可控因素导致的生产中断;

7) 车间 24 小时运转不停工.

问题描述所需的符号变量如下:

t 表示当前时间;

m 表示 *Seru* 的可用位置数目;

j 为 *Seru* 的序号索引 ($j = 1, 2, 3, \cdots$) ;

k 为车间内 *Seru* 可用位置的序号索引 ($k = 1, 2, 3, \cdots$) ;

r_j 表示 *Seru* 对应的订单任务发布时间 ($r_j \geq 0$);

p_j 表示 *Seru* 的处理时间 ($p_j > 0$);

w_j 表示 *Seru* 的权重 ($w_j \geq 0$);

S_j 表示 *Seru* 的构建时间 ($S_j \geq r_j$);

C_j 表示 *Seru* 的完工时间 ($C_j = S_j + p_j$).

决策变量:

$$X_j^k(t)=\begin{cases}1, & t\text{ 时刻 }Seru\ j\text{ 在位置 }k\text{ 上运作}\\ 0, & \text{其他}\end{cases}$$

目标量总加权完工时间的表示为:

$$\min\sum_j w_jC_j$$

根据实际生产中的问题特性和约束, 本文的约束条件描述如下:

$$S_j\geq r_j \tag{1}$$

$$\sum_{k,j}X_j^k(t)\leq m,\ \forall\, t \tag{2}$$

$$\sum_{j}X_j^k(t)\leq 1,\ \forall\, t,k \tag{3}$$

$$C_j=S_j+p_j \tag{4}$$

其中, 式 (1) 表示订单任务发布之前, 一切信息未知, 无法构建相应 $Seru$; 式 (2) 表示任意时刻工厂内最多容纳 m 个 $Seru$ 运作; 式 (3) 表示任意时刻同一位置上最多容纳一个 $Seru$ 运作; 式 (4) 表示 $Seru$ 一旦构建, 将会持续运作至完成订单任务, 中途不会被打断.

目前尚无专门针对有资源冲突的 $Seru$ 在线并行调度问题的相关研究, 考虑类似的并行机调度问题构建. $Seru$ 的空间位置对应到并行机, 将 $Seru$ 安排在相应的位置上. 并行机调度问题的跨领域应用在控制系统的故障诊断中也有体现[17−18].

对并行机调度问题, Anderson 和 Potts 率先提出了单机器下竞争比为 2 的最优确定性在线算法, 最小加权处理时间 (Shortest weighted processing time, SWPT) 算法[19]. 多机器下, Hall 等设计了一个竞争比为 $(4+\varepsilon)$ 的算法[20], 其中 ε 为任意小的正数. Megow 和 Schulz 将竞争比改进为 3.28[21]. Correa 和 Wagner 又提出了竞争比为 2.618 的无优先级 α 调度 (Non-preemptive α scheduling, NAS) 算法[22]. Sitters 设计了 ONLINE(ε) 算法[23], 并证明其竞争比不大于 $(1+1/\sqrt{m})^2\,(3\times10^{-2})/(2\times10^{-2})$, 在并行机数量较少时该算法远优于 NAS 算法, 但在并行机数量增大时 NAS 算法的竞争比趋近 1.79, 优于 ONLINE(ε) 算法. 随机化被允许时, 可以得到更好的算法, 这里不再赘述[22, 24−25].

通过推广 Megow 和 Schulz 设计的单机器下的算法[21], Tao 提出一个以总加权完工时间最小为目标的并行机在线调度算法, 即平均延迟最短加权处理时间 (Average delayed shortest weighted processing time, AD-SWPT) 算法, 算法竞争比为 $(2.5-1/(2m))$[26]. 对 AD-SWPT 进行改进, Tao 等提出一个竞争比为 $(1+(m-1+\sqrt{17m^2-2m+1})/4m)$ 的算法[27], 是目前文献中以总加权完工时间最小为目标的并行机在线调度算法中竞争比最小的一个. 但无论是 AD-SWPT 算法还是其改进算法, 所得到的竞争比都不是一个常数, 后者还极为复杂.

先基于 AD-SWPT 算法及其改进算法得到一个竞争比为常数的算法, 有利于将原本针对于并行机调度问题的算法应用于 SPS, 以及在无资源冲突的 $Seru$ 在线并行调度算法上添加冲突机制后对算法性能展开数学上的理论分析.

非常数的竞争比会让对带有资源冲突的 $Seru$ 在线并行调度算法竞争比的计算变得更加复杂, 甚至难以推进. 同时, 一个具有常数竞争比的算法也能为本领域及并行机调度等相关领域的后续研究提供更加便捷与直观的参照.

因此, 本文针对 $Seru$ 在线并行调度问题, 首先考虑无资源冲突的情况, 对 AD-SWPT 算法竞争比不为常数的局限性, 设计 α AD-SWPT 算法, 采用实例归约的方法来计算竞争比, 得到一个具有良好常数竞争比的算法, 再在 α AD-SWPT 算法的基础上, 针对带有资源冲突的 $Seru$ 在线并行调度问题, 将问题转化为计算具有特殊结构实例的竞争比, 可以得到某些特殊情况下的优化算法.

2 α AD-SWPT 在线算法

2.1 AD-SWPT 算法简介

AD-SWPT 算法. 对并行机的在线调度问题, 一旦出现空机器和一些可以被安排的任务时, 在所有已生成但未安排的任务中, 选择处理时间与权重比值 p_j/w_j (后文中用加权处理时间来代指这个比值) 最小的一个. 当出现相等的情况时, 选择处理时间较小的. 记被选择的任务为 J_i, 计算时刻 t 所有机器上正在运作的任务的剩余处理时间总和. 这个值根据表 1 中的符号可以被写作 $\sum_{S_j\leq t}\hat{p_j}(t)$. 那么如果

$$\frac{p_i+\sum\limits_{S_j\leq t}\hat{p_j}(t)}{m}\leq t \tag{5}$$

就在 t 时刻将 J_i 安排在空机器上; 否则, 等下一个时刻重复以上整个过程.

对一般情况下 AD-SWPT 算法的竞争比, 有:

定理 1[26]**.** 对以总加权完工时间为目标量的无优先级并行机在线调度问题, AD-SWPT 算法的竞争比区间为 $[2,\ 2.5-1/(2m)]$.

定理 1 的证明采用了实例归约, 该思想在对两个半在线单调度问题的分析中被首次提出[28−29]. 在

表 1　基本符号说明

Table 1　Basic symbolic explanation

符号	说明
t	当前时刻
$\hat{p}_j(t)$	对于一个可行的安排方案, 在时刻 t , 任务/ $Seru$　J_j 还剩余的处理时间, 若在时刻 t , 该任务/ $Seru$ 还未开始运作, 那么 $\hat{p}_j(t)=p_j$
$\sigma(\cdot)$	一般实例下, 应用相应算法所得到的安排方案, 也用于表示对应的目标值, 在本文中即为总加权完工时间 $\sum w_jC_j$
S_j	任务/ $Seru$　J_j 在安排方案 $\sigma(\cdot)$ 中的构建时间
C_j	任务/ $Seru$　J_j 在安排方案 $\sigma(\cdot)$ 中的完工时间
$\pi(\cdot)$	一般实例下, 应用最优离线算法所得到的安排方案, 也用于表示对应的目标值, 在本文中即为总加权完工时间 $\sum w_jC_j$

证明中发现, AD-SWPT 算法可以被进一步优化.

2.2　α AD-SWPT 算法简介

AD-SWPT 算法的竞争比 $(2.5–1/(2m))$ 不为常数, 而其优化算法的竞争比 $(1+(m-1+\sqrt{17m^2-2m+1})/4m)$, 虽然在数值上略有减小, 却更为复杂. 于是, 本文在 AD-SWPT 算法的基础上, 设计了一种竞争比为常数的调度算法. 记为 α AD-SWPT (α-average delayed shortest weighted processing time) 算法.

在式 (5) 右侧添加调节参数 α (当 $\alpha=1$ 时, 优化算法退化为 AD-SWPT 算法, $m\to\infty$ 时, 新发布的任务无需等待可立即被安排, 可以认为 $\alpha>1$ [27]), 将 t 变为 αt, 同时将并行机的调度问题, 对应到 $Seru$ 的调度问题上, 如图 1 所示. AD-SWPT 算法变化如下:

α AD-SWPT 算法. 一旦出现空位置和一些可以被构建的 $Seru$ 时, 在所有对应订单任务已发布但未构建的 $Seru$ 中选择加权处理时间 p_j/w_j 最小的一个. 当出现相等的情况时, 选择处理时间较小的. 记被选择的 $Seru$ 为 J_i , 计算时刻 t 所有位置上正在处理的 $Seru$ 的剩余处理时间总和. 这个值根据表 1 中的符号可以被写作 $\sum_{S_j\le t}\hat{p}_j(t)$. 那么如果

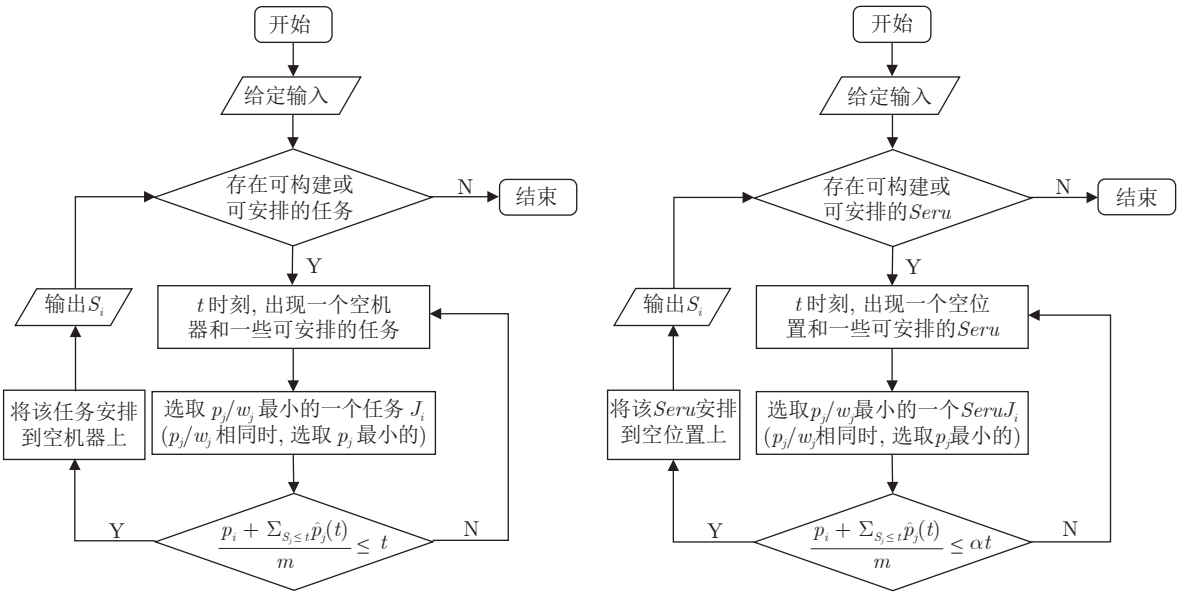
$$\frac{p_i+\sum\limits_{S_j\le t}\hat{p}_j(t)}{m}\le \alpha t \tag{6}$$

就在 t 时刻将 J_i 安排在空位置上; 否则, 等下一个时刻重复以上整个过程.

2.3　α AD-SWPT 算法竞争比的计算

尽管竞争比是指在所有可能的实例中所能达到的最坏性能比, 但穷举搜索是不可能的, 因为集合所包含的实例数目是无穷的.

性能比与竞争比的概念类似, 在接下来的证明过程中, 为了不至于造成误解, 将非一般实例集合



(a) AD-SWPT　　(b) α-SWPT

图 1　AD-SWPT 与 α AD-SWPT 算法流程图

Fig.1　The flow charts of AD-SWPT and α AD-SWPT

下的竞争比称为性能比.

α AD-SWPT 算法竞争比的计算方法与 AD-SWPT 算法同源[26], 同样采用实例归约.

实例归约的思路是在实例空间中找出最坏的情况, 证明部分实例的竞争比不小于其他实例, 从而缩小搜索空间, 在更小的集合内对算法的性能比进行分析. 这些更小集合里的实例拥有的特殊性质, 可以使对性能比计算的分析更加深入.

本文中, 最坏的情况可能从两类实例集中得到. 可以推测, 这两类实例性能比的表达式中含有调节参数 α.

2.3.1 相关定义与性质

α AD-SWPT 算法中由于存在延迟机制, 所以在一些位置上会出现空余的时间段.

为了方便叙述, 参考 AD-SWPT 算法竞争比计算中的定义[26]:

称一个位置在时刻 t 为 "闲置", 如果这个位置在时间段 $(t-\varepsilon,\ t+\varepsilon)$ 闲置; 称一个位置在时刻 t 为 "运作", 如果这个位置在时间段 $(t-\varepsilon,\ t+\varepsilon)$ 不闲置, 其中 ε 为无穷小的正数.

称时刻 t 是一个位置的 "运作起点" (记为 SPoint), 如果这个位置在时间段 $(t-\varepsilon,\ t)$ 闲置, 在时间段 $(t,\ t+\varepsilon)$ 不闲置; 称时刻 t 是一个位置的 "运作终点" (记为 EPoint), 如果这个位置在时间段 $(t-\varepsilon,\ t)$ 不闲置, 在时间段 $(t,\ t+\varepsilon)$ 闲置.

根据 Tao 在论文中的描述, 可以证明, 最坏情况下的实例在 α AD-SWPT 算法下产生的调度方案中, 所有位置最早的 SPoint 和最晚的 EPoint 之间, 不存在时刻 t, 满足所有位置都在时刻 t 闲置[26].

记满足上述性质的实例为 I_1.

$\sigma(I_1)$ 中的 $Seru$ 按照构建时间排列, 记作 J_1, $J_2, \cdots, J_n$. 以队列内 $Seru$ 的加权处理时间非减为原则, 可将 $Seru$ 分割为若干个子队列, 每个子队列最后一个 $Seru$ 的加权处理时间大于下一个子队列第一个 $Seru$ 的加权处理时间, 如图 2 所示.

2.3.2 实例归约

首先引入一个重要引理[28], 这个引理会在接下来的分析中被反复使用.

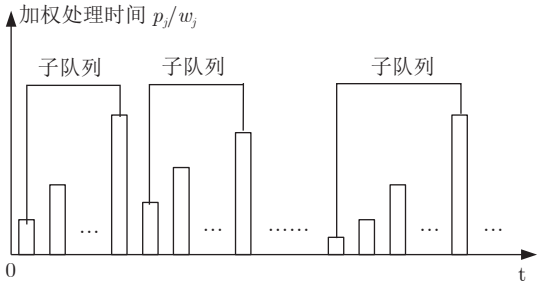


图 2 按照 α AD-SWPT 安排方案的构建时间示意图

Fig. 2 Processing sub-queues in terms of starting time in the α AD-SWPT schedule

引理 1. $f(x)$ 和 $g(x)$ 是定义在区间 $[u,v]$ 上的两个正值函数, 且 $f(x)$ 为凸函数, $g(x)$ 为凹函数. $f(x)/g(x)$ 的最大值在区间端点处取到, 即:

$$\frac{f(x)}{g(x)} \le \max\left\{\frac{f(u)}{g(u)}, \frac{f(v)}{g(v)}\right\},\ \forall x \in [u,v]$$

证明. $\forall x \in [u,v]$, $\exists a \in [0,1]$, s.t. $x = au + (1-a)v$. 又 $f(x)$ 为凸函数, $g(x)$ 为凹函数, 那么有:

$$f(x) \le af(u) + (1-a)f(v)$$

$$g(x) \ge ag(u) + (1-a)g(v)$$

又 $f(x)$ 和 $g(x)$ 是定义在区间 $[u,v]$ 上的两个正值函数, 那么有:

$$\frac{f(x)}{g(x)} \le \frac{af(u)+(1-a)f(v)}{ag(u)+(1-a)g(v)} \le \max\left\{\frac{f(u)}{g(u)}, \frac{f(v)}{g(v)}\right\}$$

可以证明, 在区间的某个端点开放时引理也成立, 此时 $f(x)/g(x)$的上确界存在于相对应的端点处.□

按照 Tao 的处理方式, 将 I_1 在性能比不减的情况下, 化归为两类特殊实例, 它们在 αAD-SWPT 算法下生成的安排方案具有更多简单特殊的子队列结构.

第一类, 实例中的 $Seru$ 拥有相同的加权处理时间, 记作 I_2; 第二类, 实例中部分 $Seru$ 的权重趋近于无穷大, 记作 I_3.

记 I_1' 为一个过渡实例, 它在 α AD-SWPT 算法下生成的安排方案满足最后一个子队列中所有 $Seru$ 具有相同的加权处理时间.

直接引用 Tao 在论文中给出的定理, 有:

表 2 无冲突时的特殊实例符号说明

Table 2 Symbolic explanation of four special instances without conflicts

符号	说明
I_1	由 α AD-SWPT 算法生成的安排方案满足在所有的位置中, 最早的 SPoint 和最晚的 EPoint 之间, 不存在任何时刻 t 所有位置都闲置的一类实例
I_1'	满足 I_1 的结构, 且满足在 α AD-SWPT 算法生成的安排方案下, 最后一个子队列中所有 $Seru$ 的加权处理时间相同的一类实例
I_2	满足 I_1 的结构, 且满足所有 $Seru$ 的加权处理时间相同的一类实例
I_3	满足 I_1 的结构, 且满足在 α AD-SWPT 算法生成的安排方案下, 最后一个子队列中所有 $Seru$ 的权重无穷大的一类实例

引理 2[26]**.** 对任意实例 I_1, 都能通过修改 I_1 中 *Seru* 的权重来找到一个过渡实例 I_1', 使得:

$$\frac{\sigma(I_1)}{\pi(I_1)} \le \frac{\sigma(I_1')}{\pi(I_1')}$$

引理 3[26]**.** 对任意实例 I_1', 都能通过修改 I_1' 中 *Seru* 的权重来构建一个实例 I_2 或 I_3, 使得:

$$\frac{\sigma(I_1')}{\pi(I_1')} \le \max\left\{\frac{\sigma(I_2)}{\pi(I_2)}, \frac{\sigma(I_3)}{\pi(I_3)}\right\}$$

2.3.3　竞争比的计算

引理 4. 在 α AD-SWPT 算法下, 对于特殊实例 I_2, 有:

$$\frac{\sigma(I_2)}{\pi(I_2)} \le \frac{5+\sqrt{17}}{4}$$

证明. 参考 Tao 在计算 AD-SWPT 算法下竞争比时的推理过程[26], 在不影响性能比大小的情况下, 对 I_2 中所有的 *Seru* 进行标准化.

记 $\sigma(I_2)$ 中最后一个 SPoint 为 r_L, 在 r_L 之后, 每一个位置上 *Seru* 都被连续安排, 不存在有位置空闲的时间段. 下面分为两种情况讨论:

1) 在 $\sigma(I_2)$ 中不存在对应订单任务先于 r_L 发布, 但于 r_L 或之后构建的 *Seru*. 将 $\sigma(I_2)$ 中于 r_L 或之后构建的 *Seru*, 按照构建时间的顺序排列, 记为 $J_1, J_2, \cdots, J_n$, 剩余 *Seru* 构成的过渡实例记为 I_2'.

记 $\sum_{S_i<r_L} \hat{p_j}(r_L) := A$, $\sum_{j=1}^{n} p_j := B$.

可以给出 $\sigma(I_2)$ 的一个放缩及 $\pi(I_2)$ 的一个下界:

$$\sigma(I_2) \le \sigma(I_2') + \left(r_L + \frac{A}{m}\right)B + \frac{B^2}{2m} + \left(1-\frac{1}{2m}\right)\sum_{j=1}^{n} p_j^2$$

$$\pi(I_2) \ge \pi(I_2') + r_L B + \frac{B^2}{2m} + \frac{1}{2}\sum_{j=1}^{n} p_j^2$$

根据 α AD-SWPT 算法, 有 $A/(\alpha m) \le r_L$. 结合以上两个式子, 有:

$$\frac{\sigma(I_2)}{\pi(I_2)} \le \max\left\{\frac{\sigma(I_2')}{\pi(I_2')}, \frac{(r_L+\frac{A}{m})B+\frac{B^2}{2m}+(1-\frac{1}{2m})\sum_{j=1}^{n} p_j^2}{r_L B + \frac{B^2}{2m} + \frac{1}{2}\sum_{j=1}^{n} p_j^2}\right\} \le \max\left\{\frac{\sigma(I_2')}{\pi(I_2')}, 1+\alpha\right\}$$

2) 在 $\sigma(I_2)$ 中存在至少一个 *Seru*, 对应订单任务于 r_L 之前发布, 但于 r_L 或之后构建, 记为 J_k.

将 $\sigma(I_2)$ 中于 r_L 或之后完成的 *Seru*, 按照构建时间的顺序排列, 记为 $J_1, J_2, \cdots, J_n$, 剩余 *Seru* 构成的过渡实例记为 I_2'.

将 $\{J_1, J_2, \cdots, J_n\}$ 分为如下两个集合:

$$Q_1 = \{J_i | S_j < r_L, C_j > r_L\} \cup \{J_k\}$$

$$Q_2 = \{J_i | S_j \ge r_L\} \setminus J_k$$

记 $\sum_{J_j\in Q_1} p_j := A$, $\sum_{J_j\in Q_2} p_j := B$.

可以给出 $\sigma(I_2)$ 的一个放缩及 $\pi(I_2)$ 的一个下界:

$$\sigma(I_2) \le \sigma(I_2') + r_L(A+B) + \frac{(A+B)^2}{2m} + \left(1-\frac{1}{2m}\right)\sum_{J_i\in Q_1\cup Q_2} p_j^2$$

$$\pi(I_2) \ge \pi(I_2') + \frac{(A+B)^2}{2m} + \frac{1}{2}\sum_{J_i\in Q_1\cup Q_2} p_j^2$$

又 $A/(\alpha m) \ge r_L$, $\sum_{J_j\in Q_1} p_j^2 \ge A^2/m$. 结合以上两个式子, 有:

$$\frac{\sigma(I_2)}{\pi(I_2)} \le \max\{\frac{\sigma(I_2')}{\pi(I_2')}, \frac{r_L(A+B)+\frac{(A+B)^2}{2m}+(1-\frac{1}{2m})\sum_{j\in Q_1\cup Q_2} p_j^2}{\frac{(A+B)^2}{2m}+\frac{1}{2}\sum_{j\in Q_1\cup Q_2} p_j^2}\} \le \max\{\frac{\sigma(I_2')}{\pi(I_2')}, 2+\frac{\frac{A(A+B)}{\alpha m}-\frac{(A+B)^2}{2m}-\frac{A^2}{2m^2}}{\frac{(A+B)^2}{2m}+\frac{A^2}{2m}}\} = \max\{\frac{\sigma(I_2')}{\pi(I_2')}, 1.5+\frac{1}{\alpha}-\frac{1}{2m}\}$$

通过将 I_2' 重写作 I_2, 不断迭代, 可以得到:

$$\frac{\sigma(I_2)}{\pi(I_2)} \le \max\left\{1+\alpha, 1.5+\frac{1}{\alpha}-\frac{1}{2m}\right\} \le \max\left\{1+\alpha, 1.5+\frac{1}{\alpha}\right\} (m\to+\infty)$$

记 $f(\alpha) = 1+\alpha$, $g(\alpha) = 1.5 + 1/\alpha$, 图像如图 3.

令 $f(\alpha) = g(\alpha)$, 可以得到:

当 $\alpha = \frac{1+\sqrt{17}}{4}$ 时, 有 $\frac{\sigma(I_2)}{\pi(I_2)} \le \frac{5+\sqrt{17}}{4} \approx 2.28$.

□

引理 5. 在 α AD-SWPT 算法下, 对于特殊实例 I_3, 有:

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \frac{5+\sqrt{17}}{4}$$

证明. 参考 Tao 在计算 AD-SWPT 算法下竞

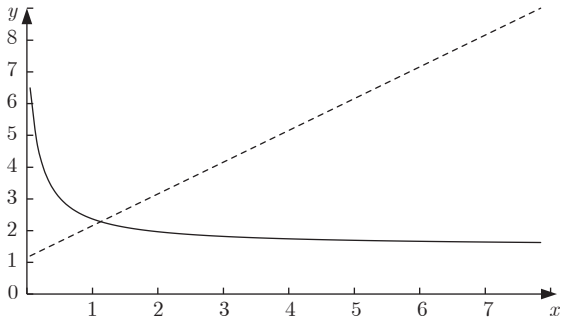


图 3 $f(\alpha)$ 与 $g(\alpha)$ 的图像

Fig. 3 Graphs of $f(\alpha)$ and $g(\alpha)$

争比时的推理过程[26], 将 $\sigma(I_3)$ 中最后一个子队列中所有 $Seru$ 的集合记为 Q_∞, 将 Q_∞ 中最早发布的订单任务的发布时间记为 r_f, 将 $\sigma(I_3)$ 中最后一个 SPoint 记为 r_L. 下面分为三种情况讨论:

1) $r_L \le r_f$.

将 Q_∞ 中的 $Seru$ 按照构建时间的顺序排列, 记为 $J_1, J_2, \cdots, J_n$. 记 $\sum_{J_j \in Q_\infty} p_j := B$. 可以得到 $\sigma(I_3)$ 的一个上界以及 $\pi(I_3)$ 的一个下界:

$$\sigma(I_3) \le \mathrm{O}(1) + \delta\left((1+\alpha) r_f B + \frac{B^2}{2m} + \left(1 - \frac{1}{2m}\right)\sum_{j=1}^{n} p_j^2\right)$$

$$\pi(I_3) \ge \mathrm{O}(1) + \delta\left(r_f B + \frac{B^2}{2m} + \frac{1}{2}\sum_{j=1}^{n} p_j^2\right)$$

其中 $\mathrm{O}(1)$ 表示一个有限的数值.

当 $\delta \to +\infty$ 时, 显然有:

$$\frac{\sigma(I_3)}{\pi(I_3)} \le 1 + \alpha$$

2) $r_L > r_f$, 且 $\sigma(I_3)$ 中所有于 r_L 时刻运作的 $Seru$ 全部属于 Q_∞. 剔除 I_3 中的某些 $Seru$, 可以得到 r_L 不是最后一个 SPoint 的过渡实例 I_3', 进而有:

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, \frac{5+\sqrt{17}}{4}\right\}$$

3) $r_L > r_f$, 且 $\sigma(I_3)$ 中于 r_L 时刻运作的 $Seru$ 中至少存在一个不属于 Q_∞. 下面再分为两种情况讨论:

a) 在 $\sigma(I_3)$ 中, Q_∞ 中不存在对应订单任务于 r_L 之前发布, 但构建于 r_L 或之后的 $Seru$. 这种情况下, 所有对应订单任务于 r_L 或之后发布的 $Seru$ 也于 r_L 或之后构建. 从 I_3 中剔除这些 $Seru$ 之后可以得到一个过渡实例 I_3'.

记 $\sum_{S_j \ge r_L} p_j := B$, 则有不等式如本页下方所示.

b) 在 $\sigma(I_3)$ 中, Q_∞ 中至少存在一个对应订单任务于 r_L 之前发布, 且构建于 r_L 或之后的 $Seru$.

将 $\sigma(I_3)$ 中, 于 r_L 时刻运作, 但不属于 Q_∞ 的 $Seru$ 组成的集合记为 Q'. 根据 α AD-SWPT 算法, 这些 $Seru$ 一定于 r_f 之前构建.

将 Q_∞ 中于 r_L 之后完成的 $Seru$ 分为如下两个集合:

$$Q_1 = \{J_i \in Q_\infty | S_j < r_L, C_j > r_L\} \cup \{J_k\}$$

$$Q_2 = \{J_i \in Q_\infty | S_j \ge r_L\} \setminus \{J_k\}$$

记 $\sum_{j \in Q'} \hat{p}_j(r_L) := A'$.

记 $\sum_{J_j \in Q_1} p_j := A$, $\sum_{J_j \in Q_2} p_j := B$.

从 I_3 中剔除掉 Q_1 和 Q_2 中的 $Seru$, 构建过渡实例 I_3'. 则有:

$$\sigma(I_3) \le \sigma(I_3') + \delta\left(\left(r_L + \frac{A'}{m}\right)(A+B) + \frac{(A+B)^2}{2m} + \left(1 - \frac{1}{2m}\right)\sum_{J_j \in Q_1 \cup Q_2} p_j^2\right)$$

$$\pi(I_3) \ge \pi(I_3') + \delta\left(r_f(A+B) + \frac{(A+B)^2}{2m} + \frac{1}{2}\sum_{J_j \in Q_1 \cup Q_2} p_j^2\right)$$

进而:

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, 1+\alpha, 1.5 + \frac{1}{\alpha} - \frac{1}{2m}\right\}$$

综合上述三种情况, 有:

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, 1+\alpha, 1.5 + \frac{1}{\alpha} - \frac{1}{2m}\right\}$$

通过将 I_3' 重写作 I_1, 不断迭代, 可以得到:

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \frac{\sigma(I_3') + \delta\left(\left(r_L + \dfrac{\sum_{S_j < r_L} p_j}{m}\right) B + \dfrac{B^2}{2m} + \left(1 - \dfrac{1}{2m}\right)\sum_{S_j \ge r_L} p_j^2\right)}{\pi(I_3') + \delta\left(r_L B + \dfrac{B^2}{2m} + \dfrac{1}{2}\sum_{S_j \ge r_L} p_j^2\right)} \le \max\left\{\frac{\sigma(I_3')}{\pi(I_3')}, 1+\alpha\right\}$$

$$\frac{\sigma(I_3)}{\pi(I_3)} \le \max\{1+\alpha, 1.5+\frac{1}{\alpha}-\frac{1}{2m}\} \le \max\{1+\alpha, 1.5+\frac{1}{\alpha}\}(m \to +\infty)$$

同引理 4 的证明, 可以得到:

当 $\alpha = \dfrac{1+\sqrt{17}}{4}$ 时, 有 $\dfrac{\sigma(I_3)}{\pi(I_3)} \le \dfrac{5+\sqrt{17}}{4} \approx 2.28.$ □

从而有如下定理:

引理 2. 对无资源冲突的 $Seru$ 在线并行调度问题, α AD-SWPT 算法的最优竞争比在 $\alpha = (1+\sqrt{17})/4$ 时取得, 为 $(5+\sqrt{17})/4 \approx 2.28$.

图 4 给出了 AD-SWPT 算法、AD-SWPT 的改进算法以及 α AD-SWPT 算法的竞争比对比图, 可以看出与 AD-SWPT 算法比较, 虽然 $Seru$ 的可用位置数目 $m = 1, 2$ 时, α AD-SWPT 算法竞争比偏大, 但是到 $m > 2$ 时, AD-SWPT 算法的竞争比不断增大, 竞争比恒为常数的 α AD-SWPT 算法的性能明显更优.

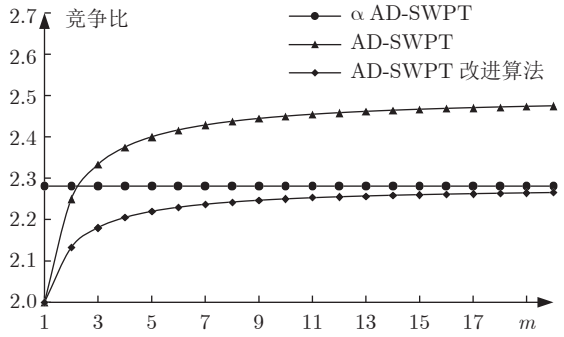


图 4　三个算法的竞争比

Fig. 4　Graphs of each algorithm′ competitive ratio

而 α AD-SWPT 算法的竞争比虽略大于 AD-SWPT 的改进算法, 不具有竞争比上的绝对优势, 但随着 m 的增大, 二者之间的细微差距在不断缩小, 直至可以忽略不计. 对此, 后文会通过计算机模拟实验在有资源冲突的情况下进行验证.

3　存在资源冲突时的特殊情况

3.1　特殊情况的介绍

若有两个 $Seru$ 由于人力资源或物力资源上的冲突, 不能同时被安排, 就称这两个 $Seru$ 是具有资源冲突的. 记本文所设计的带有资源冲突情况下的调度算法为 α AD-I (α-average delayed shortest weighted processing time-improved) 算法.

α AD-I 算法. 一旦出现空位置和一些可以被安排的 $Seru$ 时, 在所有对应订单任务已发布但未构建的 $Seru$ 中选择加权处理时间 p_j/w_j 最小的一个. 当出现相等的情况时, 选择处理时间较小的. 记被选择的 $Seru$ 为 J_i, 计算时刻 t 所有位置上正在处理的 $Seru$ 的剩余处理时间总和. 这个值根据表 1 中的符号可以被写作 $\sum_{S_j \le t} \hat{p_j}(t)$. 那么如果

$$\frac{p_i + \sum\limits_{S_j \le t} \hat{p_j}(t)}{m} \le \alpha t \tag{7}$$

并且 J_i 与正在运作的 $Seru$ 间无资源冲突, 就在 t 时刻将 J_i 安排在空位置上; 否则, 等下一个时刻重复以上整个过程.

沿用实例归约的方法, 对有资源冲突的 $Seru$ 在线并行调度问题展开讨论, 对具有特殊结构的实例进行竞争比的计算.

将 α AD-I 算法构建的 $Seru$ 中具有资源冲突的 $Seru$ 记为冲突集合 F . 若存在这样的实例, F 中先构建的 $Seru$ J_v 与后构建的 $Seru$ J_k 完工时间的比值满足 $p_v/p_k \le (1+1/m)/2$, 则记这样的实例为 I^*.

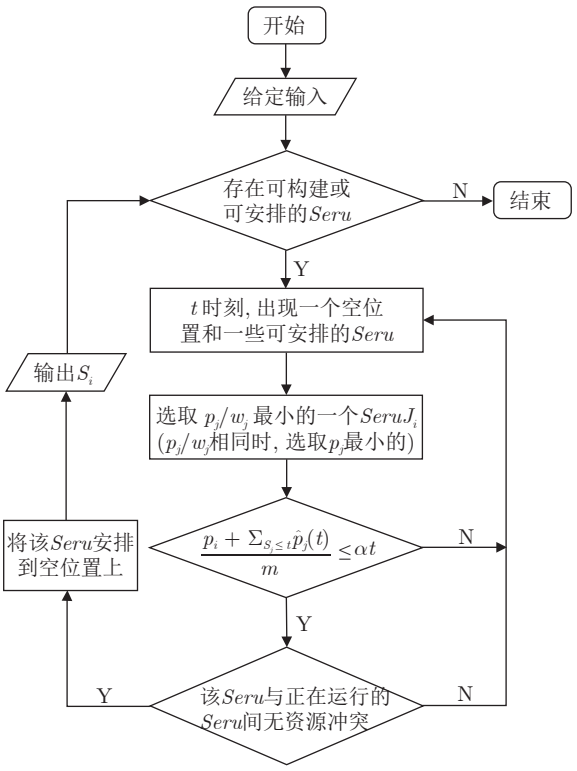


图 5　α AD-I 算法流程图

Fig. 5　The flow chart of α AD-I

3.2　α AD-I 算法竞争比的计算

显然, 对于 I^*, 第 2.3.2 节和第 2.3.3 节中的分析都是适用的. 那么有如下引理:

引理 6. 对任意实例 I^*, 都可以通过将 I^* 中部

分 $Seru$ 提出, 以及修改权重来构建出一个实例 I_2^* 或 I_3^*, 使得:

$$\frac{\sigma(I^*)}{\pi(I^*)} \leq \max\left\{\frac{\sigma(I_2^*)}{\pi(I_2^*)}, \frac{\sigma(I_3^*)}{\pi(I_3^*)}\right\}$$

引理 7. 在 α AD-I 算法下, 对于特殊实例 I_2^*, 有:

$$\frac{\sigma(I_2^*)}{\pi(I_2^*)} \leq \frac{5+\sqrt{17}}{4}$$

证明. $\sigma(I_2^*)$ 中, 同引理 4 的证明, 在不影响性能比大小的情况下, 对 I_2^* 中所有的 $Seru$ 进行标准化, 考虑 $\sigma(I_2)$ 中最后的一个 SPoint, 记为 r_L. 也就是说, 在 r_L 之后, 每一个位置上 $Seru$ 都被连续安排, 不存在有位置空闲的时间段.

引理 4 的证明分为了两种情况讨论, 显然, 情况 1) 下的证明不会发生改变.

而对情况 2), 对于对应订单任务在 r_L 之前发布, 但构建于 r_L 或之后的所有 $Seru$, 若存在一个 $Seru$ 使得式 (6) 不成立, 证明过程不会发生改变; 若对其中的任意 $Seru$, 式 (6) 均成立, 根据 α AD-I 算法, 有:

在该类子情况下, 存在一个 $Seru$, 记为 J_k, 对应订单任务于 r_L 之前发布, 但构建于 r_L 或之后, $J_k \in F$. 将 F 中于 r_L 时刻运作的 $Seru$ 记为 J_v, 则 $p_v/p_k \leq (1+1/m)/2$.

将于 r_L 或之后构建的 $Seru$, 按构建时间顺序排列, 记为 $J_1, J_2, \cdots, J_n$.

从 I_2^* 中剔除掉 $J_1, J_2, \cdots, J_n$, 记剩余的 $Seru$ 构成的过渡实例为 $I_2^{*\prime}$. $\{J_1, J_2, \cdots, J_n\}$ 对剩余 $Seru$ 的安排不会有影响.

记 $\sum_{S_i<r_L} \hat{p}_i(r_L) := A$, $\sum_{j=1}^{n} p_j := B$.

同引理 4 情况 1) 的证明, 有:

$$\sigma(I_2^*) \leq \sigma(I_2^{*'}) + \left(r_L + \frac{A}{m}\right)B + \frac{B^2}{2m} + \left(1-\frac{1}{2m}\right)\sum_{j=1}^{n} p_j^2 \tag{8}$$

将 $\{J_1, J_2, \cdots, J_n\} \setminus \{J_k\}$ 视作一个单独的实例, 且将其中所有 $Seru$ 的对应的订单任务发布时间放缩到 r_L. 同引理 4 情况 1) 的证明, 可以给出 $\pi(I_2^*)$ 的一个下界:

$$\begin{aligned}\pi(I_2^*) \geq{}& \pi(I_2^{*'}) + \pi(\{J_1, J_2, \cdots, J_n\} \setminus \{J_k\}) + (r_k + \\ & p_k)p_k \geq \pi(I_2^{*'}) + r_L(B-p_k) + \frac{(B-p_k)^2}{2m} + \\ & \frac{1}{2}\sum_{j=1}^{n} p_j^2 - \frac{1}{2}p_k^2 + (r_k+p_k)p_k\end{aligned} \tag{9}$$

又 $r_k > r_L - p_v$, 因为加权处理时间相同时会选择处理时间较小的 $Seru$ 来安排, 那么根据式 (7)、式 (8), 有不等式如本页下方所示.

再结合 $(p_k + A)/(\alpha m) \leq r_L$,以及 $p_v/p_k \leq (1+1/m)/2$. 最终有:

$$\frac{\sigma(I_2^*)}{\pi(I_2^*)} \leq \max\left\{\frac{\sigma(I_2^{*\prime})}{\pi(I_2^{*\prime})}, 1+\alpha\right\}$$

通过将 $I_2^{*\prime}$ 重写作 I_2^*, 不断迭代, 可以得到:

$$\frac{\sigma(I_2^*)}{\pi(I_2^*)} \leq \max\left\{1+\alpha, \frac{5+\sqrt{17}}{4}\right\}$$

当 $\alpha = \dfrac{1+\sqrt{17}}{4}$ 时, 有最优情况 $\dfrac{\sigma(I_2^*)}{\pi(I_2^*)} \leq \dfrac{5+\sqrt{17}}{4} \approx 2.28$. □

引理 8. 在 α AD-I 算法下, 对于特殊实例 I_3^*, 有:

$$\frac{\sigma(I_3^*)}{\pi(I_3^*)} \leq \frac{5+\sqrt{17}}{4}$$

证明. $\sigma(I_3^*)$ 中, 最后一个子队列所有 $Seru$ 的权重趋近于无穷, 同引理 5 的证明, 将 $\sigma(I_3^*)$ 中最后一个子队列中所有 $Seru$ 的集合记为 Q_∞, 将 Q_∞ 中所有 $Seru$ 间最早的订单任务发布时间记为 r_f, 将 $\sigma(I_3^*)$ 中最后一个 SPoint 记为 r_L.

引理 8 的证明分为了三种情况讨论:

1) $r_L \leq r_f$, 证明同引理 5 情况 1). 可得:

$$\frac{\sigma(I_3^*)}{\pi(I_3^*)} \leq 1+\alpha$$

2) $r_L > r_f$, 且 $\sigma(I_3^*)$ 中于 r_L 时刻运作的 $Seru$ 全部属于 Q_∞, 证明同引理 7, 从 I_3^* 剔除掉一部分 $Seru$ 后可得到 r_L 不为最后一个 SPoint 的过度实例 $I_3^{*\prime}$. 可得:

$$\frac{\sigma(I_2^*)}{\pi(I_2^*)} \leq \max\left\{\frac{\sigma(I_2^{*\prime})}{\pi(I_2^{*\prime})}, \frac{\left(r_L+\frac{A}{m}\right)B + \frac{B^2}{2m} + \left(1-\frac{1}{2m}\right)\sum_{j=1}^{n} p_j^2}{r_L(B-p_k) + \frac{(B-p_k)^2}{2m} + \frac{1}{2}\sum_{j=1}^{n} p_j^2 - \frac{1}{2}p_k^2 + (r_L - p_v + p_k)p_k}\right\} =$$

$$\max\left\{\frac{\sigma(I_2^{*\prime})}{\pi(I_2^{*\prime})}, 1 + \frac{\frac{AB}{m} + \left(\frac{1}{2}-\frac{1}{2m}\right)\sum_{j=1}^{n} p_j^2 + \frac{Bp_k}{m} - \frac{p_k^2}{2m} - \left(\frac{p_k^2}{2} - p_v p_k\right)}{r_L B + \frac{B^2}{2m} + \frac{1}{2}\sum_{j=1}^{n} p_j^2 - \frac{Bp_k}{m} + \frac{p_k^2}{m} + \left(\frac{p_k^2}{2} - p_v p_k\right)}\right\}$$

表 3　有冲突时的特殊实例符号说明

Table 3　Symbolic explanation of four special instances with conflicts

符号	说明
F	冲突集合, 实例中所有带有资源冲突的 $Seru$ 的集合
I^*	冲突集合 F 中, 先构建的 $Seru$ 与后构建的 $Seru$ 完工时间的比值总是不大于 $(1+1/m)/2$ 的一类实例
I_2^*	满足 I^* 的结构, α AD-I 算法生成的安排方案下, 最早的 SPoint 和最晚的 EPoint 之间, 不存在任何时刻 t, 所有位置都闲置; 且所有 $Seru$ 的加权处理时间相同的一类实例
I_3^*	满足 I^* 的结构, α AD-I 算法生成的安排方案下, 最早的 SPoint 和最晚的 EPoint 之间, 不存在任何时刻 t, 所有位置都闲置; 且最后一个子队列中所有 $Seru$ 的权重无穷大的一类实例

$$\frac{\sigma(I_3^*)}{\pi(I_3^*)} \le \max\left\{\frac{\sigma(I_3^{*'})}{\pi(I_3^{*'})}, 1+\alpha, 1.5-\frac{1}{2m}+\frac{1}{\alpha}\right\}$$

3)　$r_L > r_f$, 且 $\sigma(I_3)$ 中于 r_L 时刻运作的 $Seru$ 中至少存在一个不属于 Q_∞, 再分为两种情况:

a) 在 $\sigma(I_3)$ 中, Q_∞ 中不存在对应订单任务于 r_L 之前发布, 但构建于 r_L 或之后的 $Seru$. 且所有于 r_L 或之后构建的 $Seru$ 所对应的订单任务也于 r_L 或之后发布, 从 I_3 中剔除这些 $Seru$ 得到过渡实例 $I_3^{*'}$, 证明同引理 5 情况 3) a). 可得:

$$\frac{\sigma(I_3^*)}{\pi(I_3^*)} \le \max\{\frac{\sigma(I_3^{*'})}{\pi(I_3^{*'})}, 1+\alpha\} \tag{10}$$

b) 在 $\sigma(I_3)$ 中, Q_∞ 中不存在对应订单任务于 r_L 之前发布, 但构建于 r_L 或之后的 $Seru$. 但存在 $Seru$ J_k 于 r_L 之后构建但对应订单任务于 r_L 之前发布, 由 $\alpha AD-I$ 算法有 $J_k \in F$, 从 I_3^* 中剔除从 r_L 或之后构建的 $Seru$ 得到过渡实例 $I_3^{*'}$, 记 $\sum_{S_j \ge r_L} p_j := B$. 同引理 7 的证明, 有不等式如本页下方所示.

通过迭代, 最终有: $\dfrac{\sigma(I_3^*)}{\pi(I_3^*)} \le \dfrac{5+\sqrt{17}}{4}$.　□

结合引理 7 和引理 8, 对于特殊实例 I^*, 有:

定理 3. 有资源冲突的 $Seru$ 在线并行调度问题中, 对于实例 I^*, α AD-I 算法的最优竞争比在 $\alpha = (1+\sqrt{17})/4$ 时取得, 为 $(5+\sqrt{17})/4 \approx 2.28$.

4　实验与分析

4.1　特殊实例 I^* 下 α AD-I 算法的性能检测

本节对特殊实例 I^* 下 α AD-I 算法的性能进行检测, 在 NAS 算法[22] 以及 ONLINE(ε) 算法[23] 中加入与 α AD-I 算法相同的冲突处理机制, 通过计算机实验比较 I^* 下三个算法的性能.

参考 Savelsbergh 等[30] 以及 Gu[31] 在论文中采用的方法构建测试用例. 对于不属于冲突集合 F 的 $Seru$, 订单任务发布时间按照参数为 λ 的泊松分布随机生成, λ 为单位时间内平均发布的订单任务个数. 对应低、平均、高三种不同的负载, 将 λ 分别取为 0.5、1.0 和 3.0; $Seru$ 的处理时间和权重从 [1, 100] 中随机生成.

对于属于冲突集合 F 的 $Seru$, 订单任务发布时间和权重从 [1, 100] 中随机生成, 处理时间按照 I^* 的定义生成, 先构建 $Seru$ 与后构建 $Seru$ 完工时间的比值不大于 $(1+1/m)/2$.

再考虑位置数目 m, 不属于 F 的 $Seru$ 数目 n_1, 以及属于 F 的 $Seru$ 数目 n_2 之间的不同组合.

取 $m \in \{2, 5, 10, 20, 50\}$,

$n_1 \in \{m, 2m, 4m, 50, 100, 200\}$,

$n_2 \in \{2, 3, 4, 5\}$,

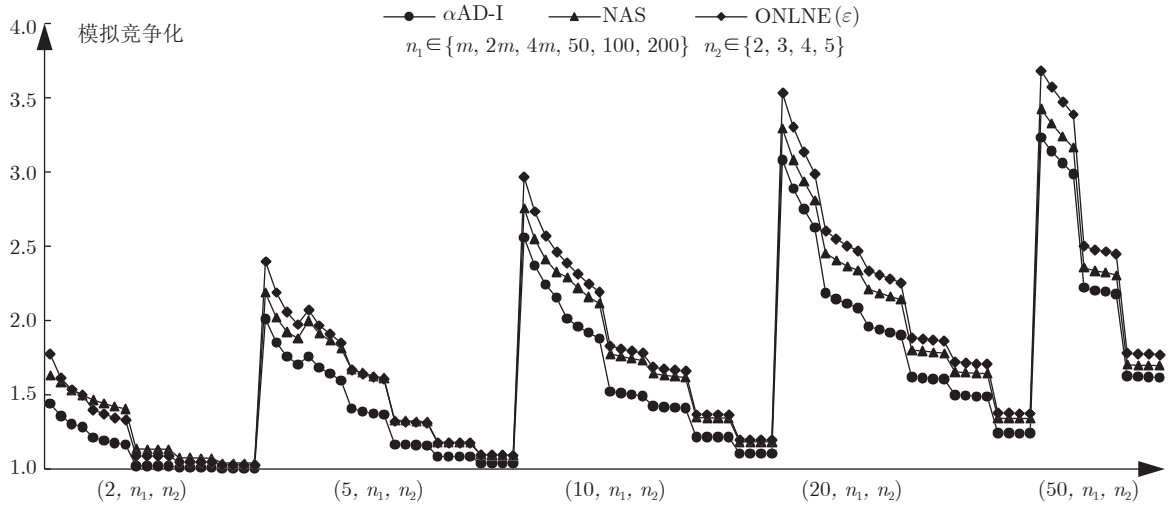
$\lambda \in \{0.5, 1.0, 3.0\}$,

且 $n_1 \ge n_2$, 则共有 312 种不同的组合, 每一种负载下有 104 种组合.

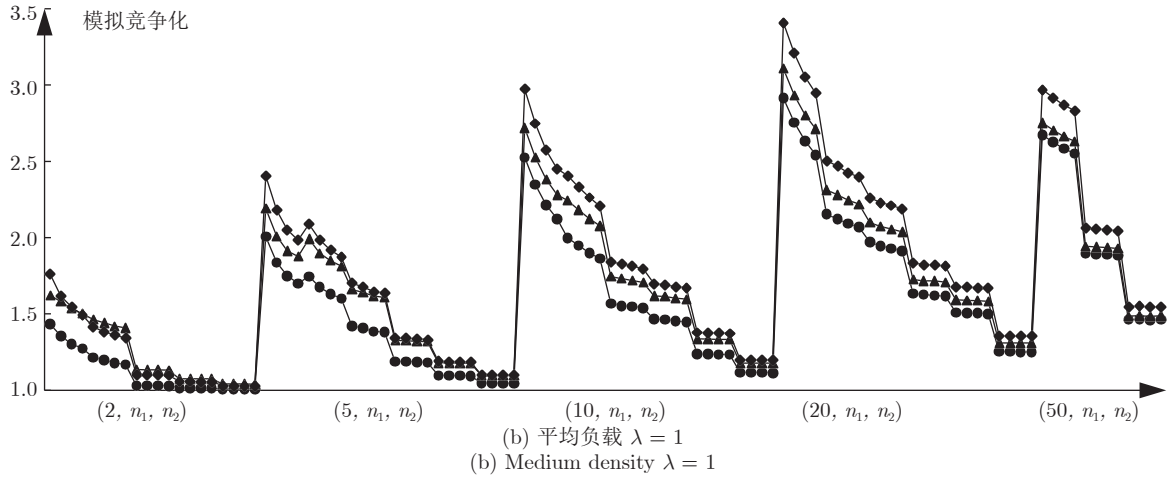
按照上述规则, 每组随机生成 1 000 个测试用例, 定义 $Seru$ 的处理时间除以 m 后单机器下最优算法 SWPT[19] 的目标值为模拟最优解, 计算 α AD-I 算法、添加冲突处理机制后的 NAS 以及 ONLINE(ε) 算法下的目标值与相应模拟最优解的比值, 称其平均值为模拟竞争比.

结果如图 6 所示, 纵坐标为模拟竞争比, 横坐标是不同的组合, 按 m、n_1、n_2 的主次排序依据升序排列.

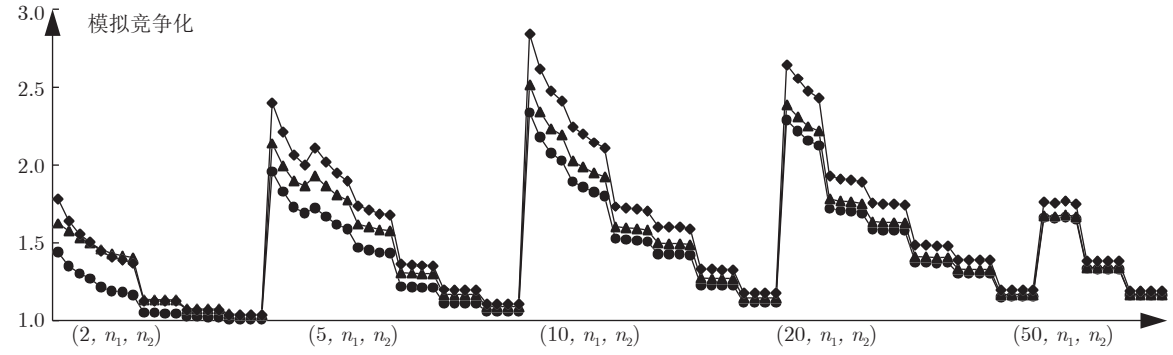
$$\frac{\sigma(I_3^*)}{\pi(I_3^*)} \le \frac{\sigma(I_3^{*'}) + \delta\left(\left(r_L + \dfrac{\sum_{S_j<r_L} p_j}{m}\right)B + \dfrac{B^2}{2m} + \left(1-\dfrac{1}{2m}\right)\sum_{S_j>r_L} p_j^2\right)}{\pi(I_3^{*'}) + \delta\left(r_L(B-p_k) + \dfrac{(B-p_k)^2}{2m} + \dfrac{1}{2}\sum_{j=1}^{n} p_j^2 - \dfrac{1}{2}p_k^2 + (r_k+p_k)p_k\right)} \le \max\left\{\frac{\sigma(I_3^{*'})}{\pi(I_3^{*'})}, 1+\alpha\right\}$$



(a) 低负载 $\lambda = 0.5$
(a) Low density $\lambda = 0.5$



(b) 平均负载 $\lambda = 1$
(b) Medium density $\lambda = 1$



(c) 高负载 $\lambda = 3$
(c) High density $\lambda = 3$

图 6 α AD-I 算法在特殊实例 I^* 下的实验结果
Fig. 6 The experimental results of α AD-I in I^*

从图 6 中可以看出, 对不同的负载, 均有特殊实例 I^* 下 α AD-I 算法的性能明显优于添加冲突机制后的 NAS 算法以及 ONLINE(ε) 算法. 且在 m 相同时, 模拟竞争比随着 n_1、n_2 的增大而减小, 算法性能与 n_1、n_2 呈正相关.

除此之外, 横向对比, 相同负载及相同 n_1、n_2 下, m 增大到一定程度时, 再继续增大, 模拟竞争比不增反减, 算法性能反而提高; 纵向对比, 相同 m、

n_1、n_2 下, 负载增大, 模拟竞争比减小, 算法性能提高. 实验结果说明在特殊实例 I^* 下, 算法对大规模定制的市场环境具有良好的适应性, 能够体现 SPS 的灵活性.

4.2　一般实例下 α AD-I 算法的性能检测

本节对一般实例下 α AD-I 算法的性能进行检测, 同样与添加冲突机制后的 NAS 算法以及 ONLINE(ε) 算法进行比较.

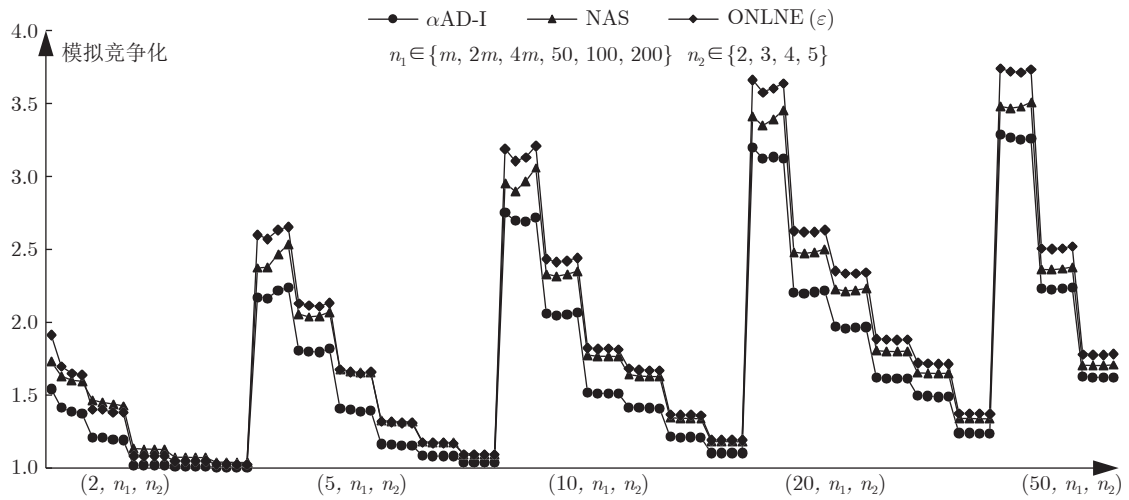
测试用例的生成与上小节类似, 唯一的区别在于, 对属于冲突集合 F 的 $Seru$, 处理时间从 [1, 100] 中随机生成. 同样对 312 种不同的组合分别随机生成 1 000 个测试用例, 计算不同算法在每种组合下的模拟竞争比.

从图 7 中可以看出, 在一般实例下 α AD-I 算法的性能也优于添加冲突机制后的 NAS 算法以及 ONLINE(ε) 算法. 且具有和特殊实例 I^* 下相似的实验结果. m 相同时, 算法性能与 n_1、n_2 呈正相关. 横向对比, m 增大到一定程度再继续增大, 算法性能反而提高; 纵向对比, 负载增大, 算法性能提高. 实验结果同样说明在市场需求大幅度波动的情况下, 算法具有良好的适应性, 能够体现 SPS 的灵活性.

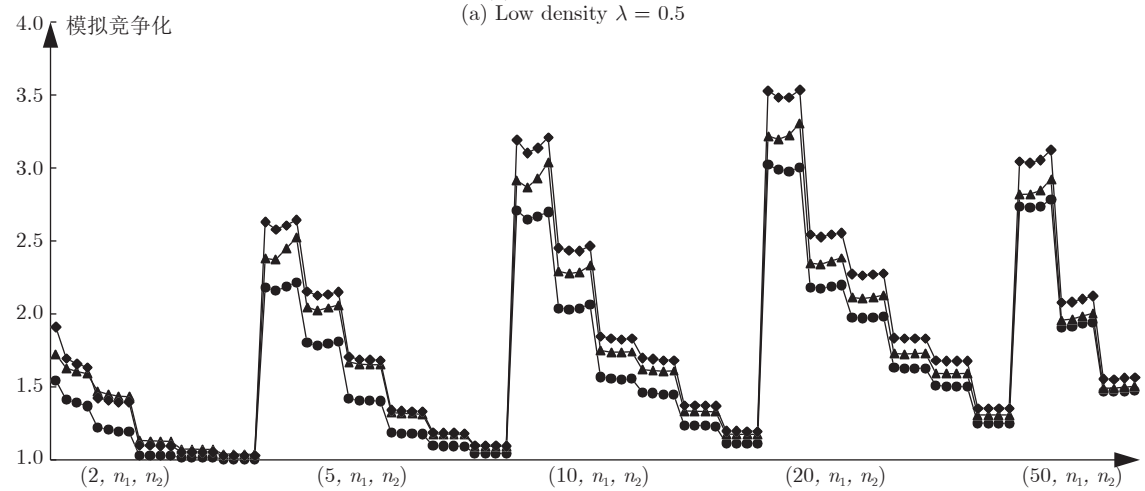
4.3　特殊实例 I^* 与一般实例下的实验结果对比

在分别对特殊实例 I^* 以及一般实例下 α AD-I 算法的性能进行检测后, 再将两种情况进行对比.

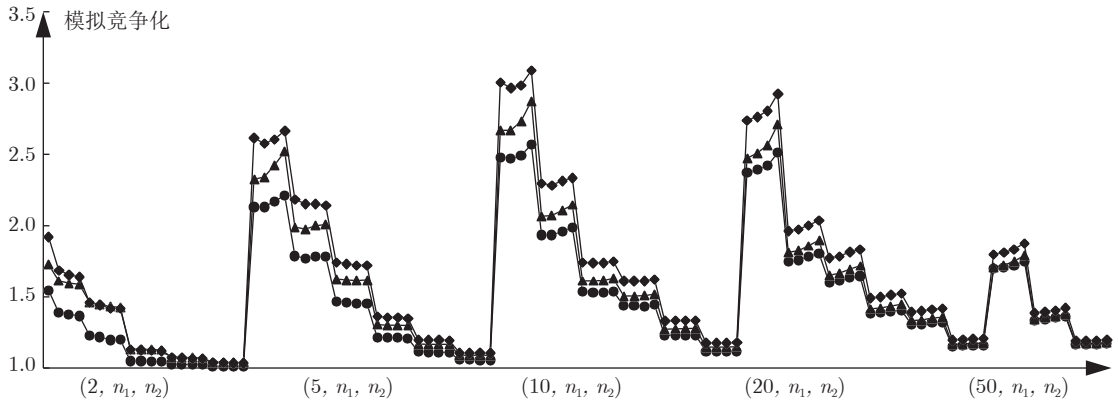
图 8 中展示了特殊实例 I^* 与一般实例下 α AD-I 算法模拟竞争比的对比, 可以看出, 相同负载下, 在 $Seru$ 的可用位置数目 m 相同时, 随着 n_1、n_2 的增大, 一般实例下的模拟竞争比与特殊实例 I^* 之间的差距不断缩小直至可以忽略. 且随着负载的增大, 这种趋势的进程在不断加快. 实验结果从一定程度上验证了, 在大规模定制的市场环境中, 一般实例下的 α AD-I 算法同特殊实例 I^* 下一样具有良好的性能.



(a) 低负载 $\lambda = 0.5$
(a) Low density $\lambda = 0.5$



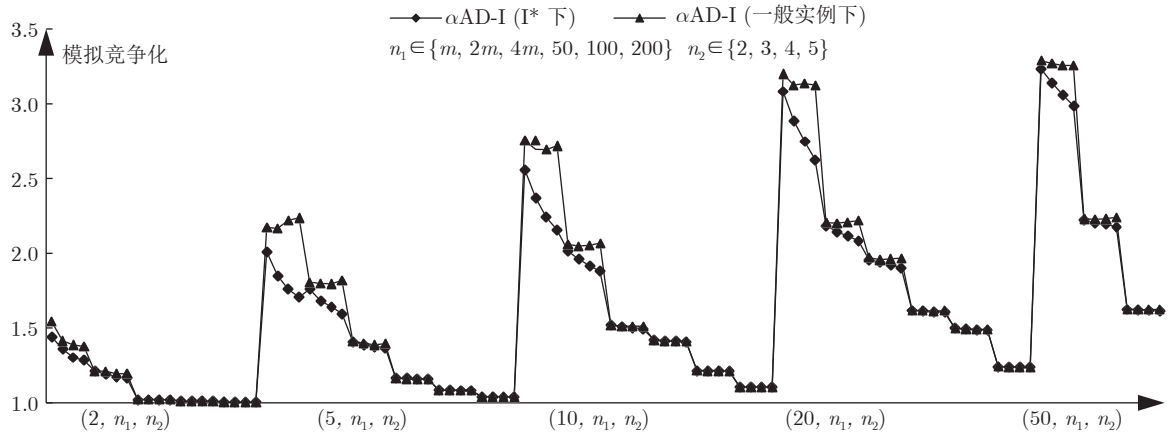
(b) 平均负载 $\lambda = 1$
(b) Medium density $\lambda = 1$



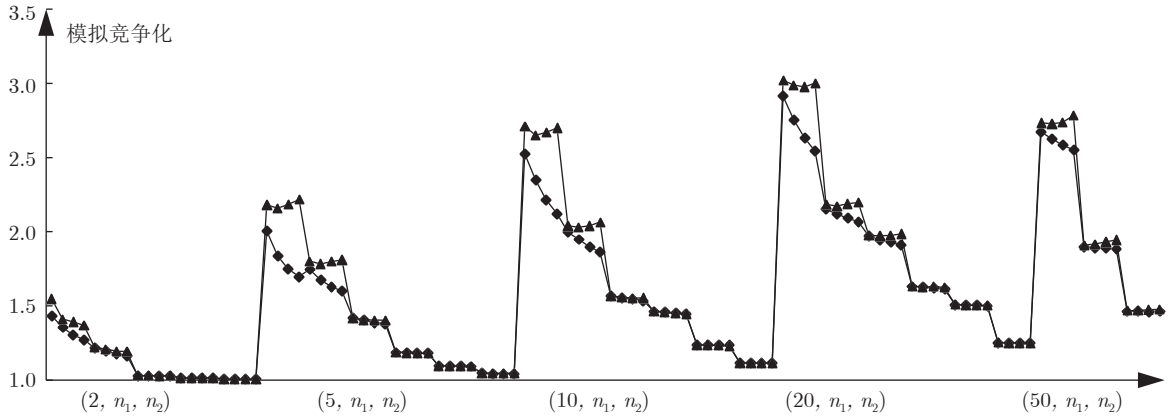
(c) 高负载 $\lambda = 3$
(c) High density $\lambda = 3$

图 7 α AD-I 算法在一般实例下的实验结果

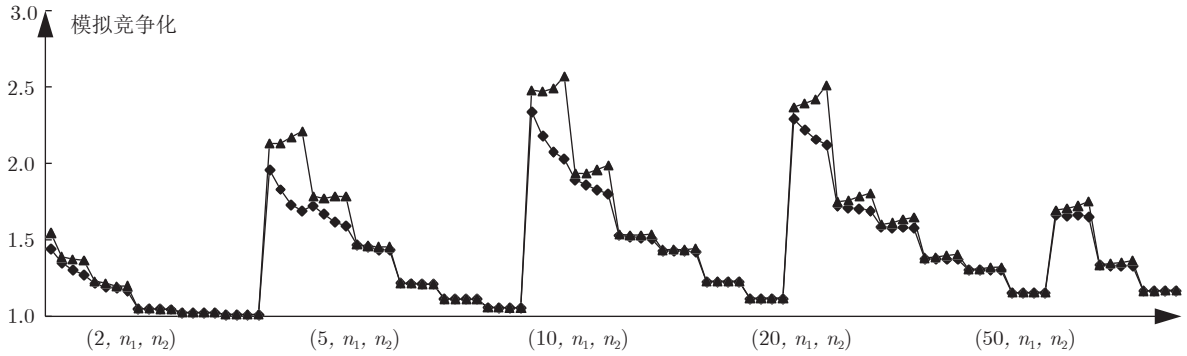
Fig. 7 The experimental results of α AD-I in general instances



(a) 低负载 $\lambda = 0.5$
(a) Low density $\lambda = 0.5$



(b) 平均负载 $\lambda = 1$
(b) Medium density $\lambda = 1$



(c) 高负载 $\lambda = 3$
(c) High density $\lambda = 3$

图 8　α AD-I 算法在 I^* 与一般实例下实验结果对比

Fig. 8　The comparision between α AD-I in I^* and α AD-I in general instances

4.4　一般实例下 α AD-I 算法与 AD-SWPT 改进算法的实验结果对比

根据第 2.3.3 节的讨论, 可知具有常数竞争比的 α AD-SWPT 算法对比 AD-SWPT 的改进算法不具有竞争比数值上的绝对优势, 但是随着 m 的增大, 二者竞争比间本就细微的差距在不断减小, 因此在大规模定制的市场环境下这种微乎其微的差距可以忽略不计.

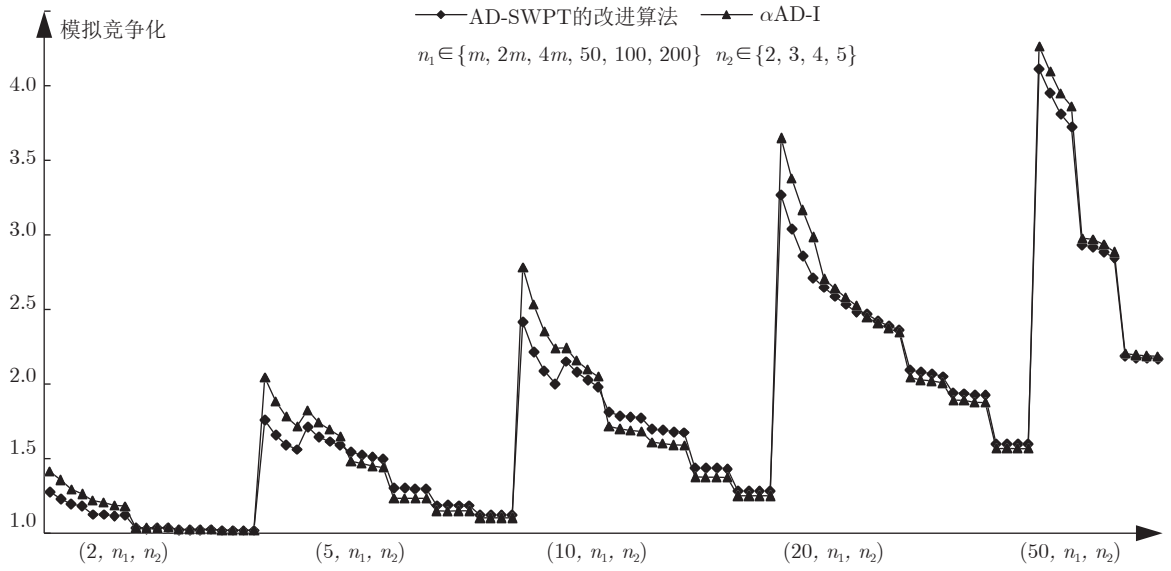
本节在一般实例下, 对基于 α AD-SWPT 算法的 α AD-I 算法与添加冲突机制后的 AD-SWPT 的改进算法进行计算机模拟实验, 测试用例的生成同第 4.2 节.

从图 9 中可以看出, 相同负载下, 在 m 相同时, 随着 n_1 与 n_2 的增大, α AD-I 算法的模拟竞争比较之添加冲突机制后的 AD-SWPT 的改进算法, 呈现从略大于到最后几乎相等的趋势, 中间甚至有 α AD-I 算法更加优越的情况出现. 且随着负载和 m 的增大, 这种趋势的进程在不断加快. $\lambda = 1$, $m = 50$ 时, 以及 $\lambda = 3$, $m = 20, 50$ 时, 两个算法的模拟竞争比曲线几乎完全重合.

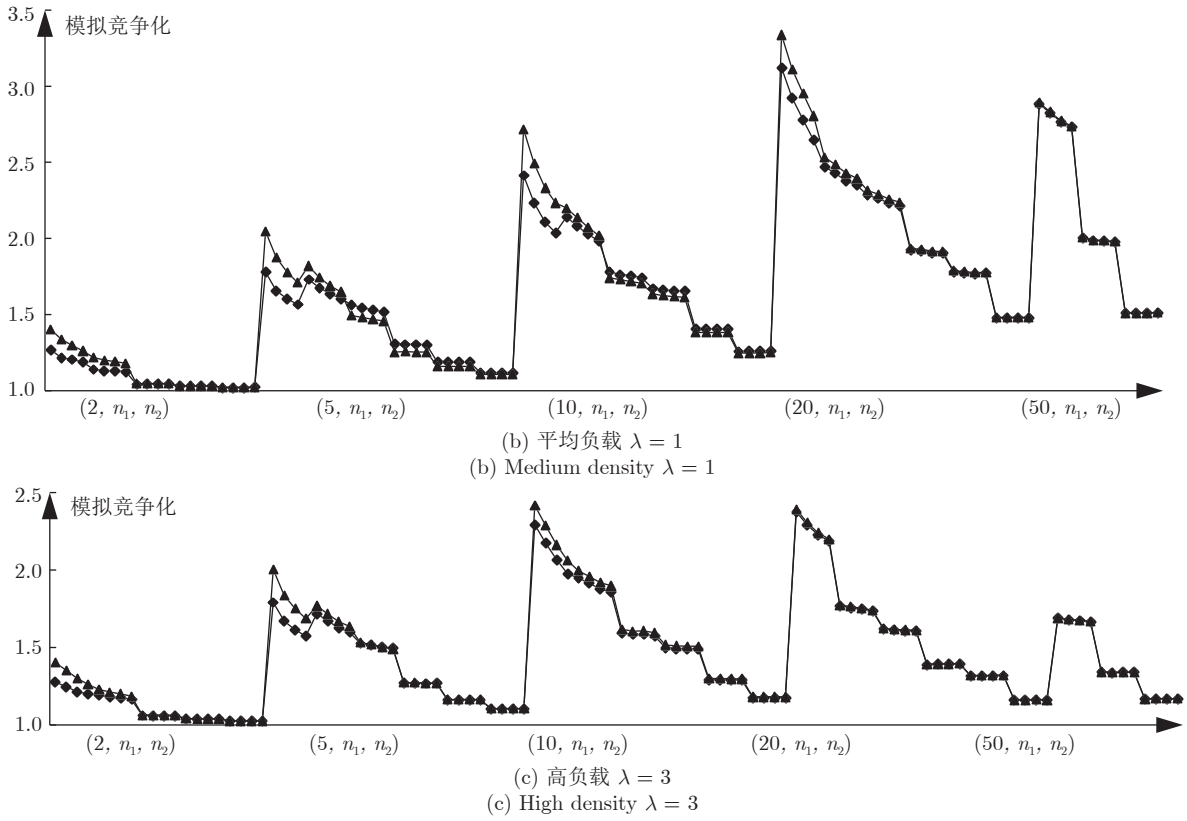
实验结果表明, 在大规模定制的市场环境下可以忽略 α AD-SWPT 算法在竞争比数值上的细微差距. 为了推进算法竞争比的计算, 在添加冲突机制时, 选用具有常数竞争比的 α AD-SWPT 算法是合理的.

5　结论

本文针对带有资源冲突的 *Seru* 在线并行调度问题, 以总加权完工时间最小为目标, 决策 *Seru* 的构建顺序及时间. 先基于并行机在线调度问题的



(a) 低负载 $\lambda = 0.5$
(a) Low density $\lambda = 0.5$



(b) 平均负载 $\lambda = 1$
(b) Medium density $\lambda = 1$

(c) 高负载 $\lambda = 3$
(c) High density $\lambda = 3$

图 9 α AD-I 算法与 AD-SWPT 改进算法在一般实例下的实验结果对比

Fig. 9 The comparision between α AD-I and improved AD-SWPT in general instances

AD-SWPT 算法, 针对无资源冲突的情况, 采用实例归约的方法, 引入调节参数 α, 得到竞争比为常数 $(5+\sqrt{17})/4 \approx 2.28$ 的 α AD-SWPT 算法.

再针对有资源冲突的情况, 基于 α AD-SWPT 算法添加冲突处理机制, 得到带有资源冲突的 *Seru* 在线并行调度算法, 即 α AD-I 算法. 沿用实例归约的方法, 在特殊实例 I^* 下可证明其竞争比同 α AD-SWPT 算法, 也为常数 $(5+\sqrt{17})/4 \approx 2.28$.

最后通过计算机模拟实验对 α AD-I 算法的性能进行检测, 对比添加冲突机制后的 NAS 算法与 ONLINE(ε) 算法, α AD-I 算法在特殊实例 I^* 与一般实例下均具有更好的性能. 且在市场需求大幅度波动的情况下, 算法具有良好的适应性, 能够发挥 SPS 灵活的特点. 且有, 在大规模定制的市场环境中, 一般实例与特殊实例 I^* 下的 α AD-I 算法均具有良好的性能.

References

1 Yin Y, Stecke K E, Li D N. The evolution of production systems from Industry 2.0 through Industry 4.0. *International Journal of Production Research*, 2018, **56**(1–2): 848–861

2 Yu Y, Sun W, Tang J F, Wang J W. Line-hybrid *seru* system conversion: Models, complexities, properties, solutions and insights. *Computers & Industrial Engineering*, 2016, **103**: 282–299

3 Roth A, Singhal J, Singhal K, Tang C S. Knowledge creation and dissemination in operations and supply chain management. *Production and Operations Management*, 2016, **25**(9): 1473–1488

4 Yin Y, Kaku I, Stecke K E. The evolution of *seru* production systems throughout Canon. *Operations Management Education Review*, 2008, **2**: 27–40

5 Liu C G, Lian J, Yin Y, Li W J. *Seru* Seisa-an innovation of the production management mode in Japan. *Asian Journal of Technology Innovation*, 2010, **18**(2): 89–113

6 Liu C G, Dang F, Li W J, Evans S, Yin Y. Production planning of multi-stage multi-option *seru* production systems with sustainable measures. *Journal of Cleaner Production*, 2015, **105**: 285–299

7 Wu Xu-Hui, Du Shao-Feng, Hao Hui-Hui, Yu Yang, Yin Yong, Li Dong-Ni. A line-*seru* conversion approach by means of cooperative coevolution. *Acta Automatica Sinica*, 2018, **44**(6): 1015–1027
(吴旭辉, 杜劭峰, 郝慧慧, 于洋, 殷勇, 李冬妮. 一种基于协同进化的流水线向*Seru*系统转化方法. 自动化学报, 2018, **44**(6): 1015–1027)

8 Yin Y, Stecke K E, Swink M, Kaku I. Lessons from *seru* production on manufacturing competitively in a high cost environment. *Journal of Operations Management*, 2017, **49–51**: 67–76

9 Stecke K E, Yin Y, Kaku I. *Seru* production: An extension of just-in-time approach for volatile business environments. *Analytical Approaches to Strategic Decsion-Making: Inter-disciplinary Considerations*, IGI Global, 2014, 45–58

10 Isa K, Tsuru T. Cell production and workplace innovation in Japan: Toward a new model for Japanese manufacturing. *Industrial Relations: A Journal of Economy and Society*, 2002, **41**(4): 548–578

11 Stecke K E, Yin Y, Kaku I, Murase Y. *Seru*: The organizational extension of JIT for a super-talent factory. *International Journal of Strategic Decision Sciences*, 2012, **3**(1): 106−119

12 Liu C G, Yang N, Li W J, Lian J, Evans S, Yin Y. Training and assignment of multi-skilled workers for implementing *seru* production systems. *The International Journal of Advanced Manufacturing Technology*, 2013, **69**(5−8): 937−959

13 Yu Y, Tang J F, Gong J, Yin Y, Kaku I. Mathematical analysis and solutions for multi-objective line-cell conversion problem. *European Journal of Operational Research*, 2014, **236**(2): 774−786

14 Yu Y, Tang J F, Sun W, Yin Y, Kaku I. Combining local search into non-dominated sorting for multi-objective line-cell conversion problem. *International Journal of Computer Integrated Manufacturing*, 2013, **26**(4): 316−326

15 Jia Ling-Yun, Li Dong-Ni, Tian Yun-Na. An intercell scheduling approach using shuffled frog leaping algorithm and genetic programming. *Acta Automatica Sinica*, 2014, **40**(5): 936−948 (贾凌云, 李冬妮, 田云娜. 基于混合蛙跳和遗传规划的跨单元调度方法. 自动化学报, 2014, **40**(5): 936−948)

16 Tian Yun-Na, Li Dong-Ni, Li Zhao-He, Zheng Dan. A hyper-heuristic approach with dynamic decision blocks for inter-cell scheduling. *Acta Automatica Sinica*, 2016, **42**(4): 524−534 (田云娜, 李冬妮, 刘兆赫, 郑丹. 一种基于动态决策块的超启发式跨单元调度方法. 自动化学报, 2016, **42**(4): 524−534)

17 Wu Y K, Jiang B, Lu N Y. A descriptor system approach for estimation of incipient faults with application to high-speed railway traction devices. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017, **49**(10): 2108−2118

18 Wu Y K, Jiang B, Wang Y L. Incipient winding fault detection and diagnosis for squirrel-cage induction motors equipped on CRH trains. *ISA Transactions*, 2020, **99**: 488−495

19 Anderson E J, Potts C N. Online scheduling of a single machine to minimize total weighted completion time. *Mathematics of Operations Research*, 2004, **29**(3): 686−697

20 Hall L A, Schulz A S, Shmoys D B, Wein J. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 1997, **22**(3): 513−544

21 Megow N, Schulz A S. On-line scheduling to minimize average completion time revisited. *Operations Research Letters*, 2004, **32**(5): 485−490

22 Correa J, Wagner M. LP-based online scheduling: From single to parallel machines. *Mathematical Programming*, 2009, **119**(1): 109−136

23 Sitters R. Efficient algorithms for average completion time scheduling. *Integer Programming and Combinatorial Optimization*. Berlin: Springer Berlin Heidelberg, 2010. 411−423

24 Chakrabarti S, Phillips C A, Schulz A S, Shmoys D B, Stein C, Wein J. Improved scheduling algorithms for minsum criteria. *Automata, Languages and Programming*. Berlin: Springer Berlin Heidelberg, 1996. 646−657

25 Schulz A S, Skutella M. Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 2002, **15**(4): 450−469

26 Tao J P. A better online algorithm for the parallel machine scheduling to minimize the total weighted completion time. *Computers and Operations Research*, 2014, **43**: 215−224

27 Tao J P, Huang R, Liu T. A 2.28-competitive algorithm for online scheduling on identical machines. *Journal of Industrial and Management Optimization*, 2015, **11**(1): 185−198

28 Tao J P, Chao Z J, Xi Y G. A semi-online algorithm and its competitive analysis for a single machine scheduling problem with bounded processing times. *Journal of Industrial and Management Optimization*, 2010, **6**(2): 269−282

29 Tao J P, Chao Z J, Xi Y G, Tao Y. An optimal semi-online algorithm for a single machine scheduling problem with bounded processing time. *Information Processing Letters*, 2010, **110**(8-9): 325−330

30 Savelsbergh M W P, Uma R N, Wein J. An experimental study of lp-based approximation algorithms for scheduling problems. *INFORMS Journal on Computing*, 2005, **17**(1): 123−136

31 Gu H Y. Computation of approximate α-points for large scale single machine scheduling problem. *Computers and Operations Research*, 2008, **35**(10): 3262−3275

江煜舟 北京理工大学计算机学院博士研究生. 主要研究方向为赛如生产智能优化.
E-mail: jiang_yuzhou@163.com
(**JIANG Yu-Zhou** Ph. D. candidate at the School of Computer Science, Beijing Institute of Technology. Her research interest covers *seru* production and intelligent optimization.)

李冬妮 北京理工大学计算机学院教授. 主要研究方向为智能优化与仿真计算, 智慧工厂与数字孪生. 本文通信作者.
E-mail: ldn@bit.edu.cn
(**LI Dong-Ni** Professor at the School of Computer Science, Beijing Institute of Technology. Her research interest covers intelligent optimization and simulation, smart factory and digital twin. Corresponding author of this paper.)

靳洪博 北京理工大学计算机学院博士研究生. 主要研究方向为赛如生产智能优化.
E-mail: hb@bit.edu.cn
(**JIN Hong-Bo** Ph. D. candidate at the School of Computer Science, Beijing Institute of Technology. His research interest covers *seru* production and intelligent optimization.)

殷 勇 同志社大学商学院教授. 主要研究方向为赛如生产与工业 4.0.
E-mail: yyin@mail.doshisha.ac.jp
(**YIN Yong** Professor at the Graduate School of Business, Doshisha University. His research interest covers *seru* production and Industry 4.0.)