计算机科学
COMPUTER SCIENCE

云计算环境下多截止期工作调度算法研究

刘志民, 陈建二

引用本文

刘志民, 陈建二. 云计算环境下多截止期工作调度算法研究[J]. 计算机科学, 2024, 51(11A): 240100120-7.

LIU Zhimin, CHEN Jianer. Scheduling Jobs with Multiple Deadlines in Cloud[J]. Computer Science, 2024, 51(11A): 240100120-7.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240100120

云计算环境下多截止期工作调度算法研究

刘志民　陈建二
广州大学计算机科学与网络工程学院　广州 510006
(liureka@163.com)

摘　要　随着大数据对人们生活的影响逐渐增大,数据存储和计算需求不断增加,云计算的兴起有效地满足了这一需求。在实时性要求较高的云计算系统中,来自客户端的资源请求被视为具有截止期限和一定收益的两阶段工作,云服务器被视为两阶段机器。不同资源请求的截止期限通常不同,如果云中心能在资源请求的截止期限之前完成该请求,就可以获得相应的收益。现有的以收益最大化作为优化目标的两阶段工作调度均是在一个公共截止期限制下进行的,而实际情况往往是不同的资源请求可能有不同的截止期限。基于当前云计算应用和数据中心数据处理的需求,建立了云计算系统中工作调度的新数学模型。首次提出了具有多个截止期的两阶段工作在多处理机上的调度问题,并给出了一个近似比为$(3k+\epsilon)$的多项式时间近似算法。当机器数目为固定常数时,近似比进一步降低为$(k+\epsilon)$,其中k是一个固定常数,即截止期的个数,ϵ是大于0的任意常数。针对特殊的T-处理时间大于R-处理时间模型,在单个两阶段机器上,给出了一个近似比为2的伪多项式时间近似算法,进一步降低了算法的近似比。

关键词:云计算;多阶段工作;多处理机调度;近似算法;算法分析
中图分类号　TP301.6

Scheduling Jobs with Multiple Deadlines in Cloud

LIU Zhimin and CHEN Jianer
School of Computer Science and Cyber Engineering,Guangzhou University,Guangzhou 510006,China

Abstract　With the increasing impact of big data on people's lives,the demand for data storage and computation continues to grow,and the emergence of cloud computing effectively meets this demand. In cloud computing systems with high real-time requirements,resource requests from clients are regarded as 2-stage jobs with deadlines and certain profits,while cloud servers are seen as 2-stage machines. The deadlines for different resource requests are usually different,and if the cloud center can complete the request before the deadline,it can obtain corresponding profits. Existing 2-stage jobs scheduling algorithms that aim to maximize revenue are conducted under a common deadline constraint,whereas in reality,different resource requests may have varying deadlines. Based on the demand in the research and applications in cloud computing and data centers,we build a mathematical model for job scheduling in cloud computing. We study the problem of scheduling 2-stage jobs with multiple deadlines on multiple 2-stage machines. Let k be the number of deadlines of the jobs. When k is a constant,a polynomial-time approximation algorithm with approximation ratio$(3k+\epsilon)$ is provided. When the number of machines is a fixed constant,the approximation ratio is further improved to$(k+\epsilon)$,where $\epsilon > 0$ is an arbitrary constant. Therefore,when k is a constant,the problem has a constant ratio polynomial-time approximation algorithm. In the case where T-processing time is greater than R-processing time,a pseudo-polynomial time approximation algorithm with approximation ratio 2 is presented,further improving the approximation ratio.

Keywords　Cloud computing,Multiple-stage jobs,Multiple machinescheduling,Approximation algorithms,Algorithm analysis

1　引言

随着大数据时代的到来,各行业对数据存储资源和计算资源的需求与日俱增,从而对现有存储和计算资源整合的需求变的越来越迫切。云计算技术的出现很好地满足了这种需求。云计算以网络作为服务交付的形式,对用户提供服务于数据中心的硬件和系统软件[1]。云中的现代数据中心通常包含大量的服务器。这些服务器由CPU、内存、网络接口和本地高速I/O组成,应用程序和数据作为资源存储在这些服务器中。在云计算环境下,客户以服务的形式动态地请求所需资源[2]。相比传统的方式,云计算技术可以有效地管理和整合零散的、异构的资源,统一对资源进行分配,提高资源利用率。

本文对云计算环境下带多个截止期的多机调度问题的研究源于目前云计算技术和数据中心的需求。近年来,清华大学张尧学院士团队提出了一个名为TransOS的云范例[3],它不仅以应用软件和数据为资源,而且还以操作系统等系统软件为资源。根据张尧学院士团队对透明计算的最新研究[4],操作系统等资源存储在服务器上,并以透明的方式按需传输到终端设备,在终端设备上进行计算。在这样的系统中,客户

通信作者:陈建二(jianer@gzhu.edu.cn)

端请求的大量资源是系统/应用软件的代码,这些代码通常体积较大,需要存储在辅助存储器中。因此,系统中的客户端设备可以非常轻,并且非常多样化。对于对实时性有较高的要求客户端,其资源请求还可能会有一个截止期,必须在截止期之前完成,才能满足客户端的需求。当服务器收到来自客户端的资源请求时,服务器首先需要将资源从辅助存储器读取到主存储器,然后通过网络将资源发送给客户端。因此,资源请求由磁盘读取操作和网络传输操作两个阶段的工作组成。每个请求在服务器中都要经过这两个阶段的操作,而网络传输操作必须在所请求的资源(数据或应用程序)由辅助存储器读入主存储器之后才能启动。服务器中磁盘读取操作时间和网络传输操作时间两者都不能被简单地忽略[5]。

由此我们抽象出本文的调度模型:一个客户端请求可以被看作是一个具有截止期、有两个需要完成的操作阶段的两阶段工作 J,称之为 R-阶段和 T-阶段,分别对应工作 J 的磁盘读取操作和网络传输操作。一个云中心的服务器可被看作是一个两阶段机器 M,由可以并行运行的 R-处理器和 T-处理器组成。工作 J 中两个阶段的操作必须在同一个机器 M 上完成,并要求只有完成 R-阶段操作后才能够开始执行 T-阶段操作。每个工作 J 还带有一个截止期 d 和一个收益值 p,要求只有在截止期 d 之前完成工作 J 才能获得收益 p。给定一组两阶段工作,在两阶段机器上对它们进行调度,使完成请求后获得的收益最大化,这正是本文研究的调度模型。

2 相关工作

调度是运筹学研究中最广泛的领域之一。一般调度问题可以描述如下:在尽可能满足约束条件(如截止期、处理路线和资源情况等)的前提下,通过对安排工作的处理时间和处理顺序以及使用哪些资源进行探索,以获得工作的执行时间或成本的最优化[6]。Flow-shop 调度是许多调度问题的简化模型,与实际生产联系极为密切,是目前研究较为广泛的一种调度模型。Flow-shop 调度模型的早期研究集中在最小化完成时间的优化目标上。经典的 Johnson 算法[7]表明在单个两阶段机器上调度两阶段工作以最小化完成时间的调度问题可以在时间 $O(n\log n)$ 内解决,其中 n 是工作的个数。关于多个两阶段机器上的调度问题,由于经典的 NP-难问题 Makespan[8]可以被看作是两阶段工作调度问题的化简,因此即使机器的数量 m 是固定的,两阶段工作调度问题也是 NP-难的。除非 P=NP,否则没有算法能在多项式时间内产生两阶段工作调度问题的最优解。

因此,研究两阶段工作调度问题的算法,主要围绕着设计多项式时间的近似算法。Kovalyov[9]率先研究了以最小化完成时间(Makespan)为优化目标的多个两阶段机器上的调度问题。基于与文献[9]中类似的公式,Dong 等[10]对此问题提出了一种新的方法,给出了一个在机器数量 m 为固定常数时的伪多项式时间算法,并构造了一个完全多项式时间的近似方案。Wu 等[11]提出了与文献[9-10]完全不同的新公式,同样给出了机器数量为固定常数 m 时的完全多项式时间近似方案,并显著缩短了算法的运行时间。Wu 等[12]还研究了以机器数量 m 为输入时的两阶段工作调度问题。在这种情况下,问题被证明是强 NP-难的。Wu 等考虑了该问题的两种限制情况:一种是限制每个工作的 R-操作比 T-操作更耗时,而另一种是限制每个工作的 T-操作比 R-操作更耗时,对于第一种情况,Wu 等提出了在线 2 竞争比算法和离线 11/6 近似比算法;对于第二种情况,Wu 等提出了在线 5/2 竞争比算法和离线 11/6 近似比算法。对于一般的情况,Wu 等还给出了近似比为 2.6 的近似算法[13]。在进一步的工作中,Wu 等将一般情况和先前引入的特殊情况的近似比分别从 17/6 和 11/6 提高到 $(2+\epsilon)$ 和 $(1.5+\epsilon)$,其中 $\epsilon > 0$ 可以是任意常数[14]。最近,Dong 等给出了该问题的多项式时间近似方案[15]。

虽然目前大多数研究关注的是以最短完成时间为优化目标的两阶段工作调度问题,但是 Conway 等提出,在实践中,人们倾向于关注具有截止期的两阶段工作调度问题[16]。有一个公共截止期的单阶段工作在单机上的调度问题,其实就是经典的背包问题[8]:在背包问题实例中,一个大小为 s、利润为 p 的物品 (s,p) 可以被看作是一个时间为 s、利润为 p 的单阶段工作,而一个容量为 D 的背包可以被看作是一个有公共截止期 D 的单阶段机器。因此,可以将多背包问题看作是在多个单阶段机器上调度有一个公共截止期的单阶段工作问题。这个问题近年来引起了研究者的广泛兴趣,并取得了一些深刻而有意义的成果[17]。Chen 等[18]根据实际生产的需要,研究了经典的背包和装箱问题的推广,并提出了一种参数化近似算法。

针对有单个公共截止期的两阶段工作的调度问题,近年来进行了一些研究。如 Croce 等[19]研究了有公共截止期的两阶段工作调度问题的未加权版本,并提供了枚举式计算程序。Dawande 等[20]研究了单机器上有公共截止期的两阶段工作调度问题,提出了近似算法和启发式算法,并给出了启发式算法的实验结果。Chen 等[21]针对两阶段机器的数量是输入的一部分的情况,结合多重背包问题近似算法研究的最新进展,提出了一个近似比为 3 的近似算法,针对两阶段机器的数目为固定常数的情况,提出了一个近似比为 2 的多项式时间近似算法。在单个公共截止期的两阶段工作的调度问题上,最好的结果是由 Tong 等在 2022 年得到的多项式时间近似方案[22]。

在本研究之前,还未有针对具有多个截止期的两阶段工作在多个机器上的调度问题的研究。由于该问题是由云计算应用引发的,对应于客户购买云服务的云计算的服务模式,是云计算应用中必须解决的问题。这就是本文研究的应用背景。

3 问题建模

3.1 相关术语和符号

两阶段工作(简称工作):由 4 个整数组成的元组表示,$J=(r,t,d,p)$。其中 r 和 t 分别表示工作 J 的 R-阶段处理时间与 T-阶段处理时间,d 表示工作 J 的截止期,p 表示工作 J 在不大于时间 d 之前完成所获得的收益。

工作集合 $G=\{J_1,J_2,\cdots,J_n\}$:由若干个工作组成的集合。

k-截止期集合 $D_k=\{T_1,T_2,\cdots,T_k\}$:指在给定的工作集

G 所有的工作中，共有 k 个不同的截止期 $T_1, T_2, \cdots, T_k$。

两阶段机器 M（简称机器）：具有两个并行操作处理器，即 R-处理器与 T-处理器。我们假设所有每个机器都是相同的。

工作序列 $S=\langle J_a, J_b, \cdots, J_z\rangle$（调度 S）：指工作 $J_a, J_b, \cdots, J_z$ 按所给次序在一个机器 M 上依次执行。如果工作序列 S 中的每一个工作均能在不超过其截止期前完成，则称工作序列 S 为机器 M 上的一个可行解。工作序列 S 的开始时刻指序列中第一个工作 J_a 的 R-处理开始时刻，结束时刻指序列中最后一个工作 J_z 的 T-处理结束时刻。工作序列 S 的完工时间（makespan）指工作序列 S 结束时刻与开始时刻之间的时间差值。

收益函数 P：表示一个工作集或者一个工作序列中的工作收益之和。例如，对于一工作集合 $G=\{J_i=(r_i,t_i,d_i,p_i)\mid 1\leqslant i\leqslant n\}$，则 $P(G)=\sum_{i=1}^{n}p_i$。

符号◇表示调度连接。调度 S_i 与调度 S_j 连接后得到 $(S_i\diamond S_j)$，表示将调度 S_j 中的工作按先后顺序逐一追加到调度 S_i 的末尾，即需要满足两阶段工作在两阶段机器上的执行规则。

符号＋表示调度拼接。调度 S_i 与调度 S_j 拼接后得到调度 (S_i+S_j)，表示调度 S_j 的开始时刻等于调度 S_i 的结束时刻。注意在调度 (S_i+S_j) 中，S_i 的最后一个工作与 S_j 的第一个工作之间的 R-处理不是连续执行的。不难发现，(S_i+S_j) 的完工时间等于调度 S_i 的完工时间与调度 S_j 的完工时间之和。

3.2 问题定义

如果一个两阶段工作集合 G' 中的工作可以分解成 m 个工作序列 $\langle S_1', S_2', \cdots, S_m'\rangle$，使得每个工作序列 S'_h $(1\leqslant h\leqslant m)$ 是一个两阶段机器的可行解，则称集合 G' 为 m-机器的一个可行集合，工作序列集合 $\langle S_1', S_2', \cdots, S_m'\rangle$ 则称为一个可行解，或可行调度。

本文研究的主要问题定义如下：

收益最大化的多机器多截止期两阶段工作调度问题（Multi-Flowshop Scheduling with Maximized Profit under Multi-Deadlines，$\mathrm{MFPD}_k(G,D_k,m)$）：给定一个两阶段工作集合 G，k-截止期集合 D_k 和 m 个两阶段机器，寻找工作集合 G 的一个子集 G' 以及由 G' 中工作组成的工作序列集合 $S'=\langle S_1', S_2', \cdots, S_m'\rangle$，满足以下条件：1）$G'$ 为 m-机器的一个可行集合；2）工作序列集合 $\langle S_1', S_2', \cdots, S_m'\rangle$ 为 G' 在 m-机器上的一个可行解；3）G' 为所有 G 中 m-机器可行子集中收益最大的可行子集。称工作集合 G 中这样的可行子集 G' 及其组成的可行工作序列集合 S' 为 $\mathrm{MFPD}_k(G,D_k,m)$ 问题在工作集合 G 上的一个最优解或最优调度，其对应的收益 $P(G')$ 为问题的最优解值。

Wu 等[13]指出，在一些应用中，web 服务有很高的磁盘 I/O 速率。在这种情况下，磁盘读取（R-处理）时间总是比网络传输（T-处理）时间少。基于这一特殊情况，我们定义一种特殊的 MFPD_k 模型，记为 $\mathrm{MFPD}_{k,R\leqslant T}(G,D_k,m)$ 模型，其中工作集合 G 中的每个工作 $J=(r,t,d,p)$ 均满足条件 $r\leqslant t$。

当工作集合 G 中所有的工作具有相同的截止期 T_1，则问题 $\mathrm{MFPD}_{k,R\leqslant T}(G,D_k,m)$ 变为问题 $\mathrm{MFPD}_{1,R\leqslant T}(G,D_1,m)$。文献[9]证明了 $\mathrm{MFPD}_{1,R\leqslant T}(G,D_1,m)$ 问题是 NP-难的。因为问题 $\mathrm{MFPD}_{1,R\leqslant T}(G,D_1,m)$ 是问题 $\mathrm{MFPD}_{k,R\leqslant T}(G,D_k,m)$ 和问题 $\mathrm{MFPD}_k(G,D_k,m)$ 的子问题，所以问题 $\mathrm{MFPD}_{k,R\leqslant T}(G,D_k,m)$ 和问题 $\mathrm{MFPD}_k(G,D_k,m)$ 也均是 NP-难的。

4 MFPD_k 模型算法设计

4.1 最优解分析

在 $\mathrm{MFPD}_k(G,D_k,m)$ 模型下，根据 D_k 中的 k 个截止期 $T_1, T_2, \cdots, T_k$ 将工作集 G 划分为 k 个子集：$G=\{G_1\cup G_2\cup\cdots\cup G_k\}$，使得 G_i 中工作的截止期是 T_i $(1\leqslant i\leqslant k)$。假设 $T_1<T_2<\cdots<T_k$ 且 k 是一个常数。为方便调度算法的设计，下面定义一些符号。

令 $S^*=\langle S_1^*, S_2^*, \cdots, S_m^*\rangle$ 为工作集合 G 在 m-机器上的一个最优解，$opt=P(S^*)$ 为其对应的最优解值，其中 S_h^* $(1\leqslant h\leqslant m)$ 为机器 M_h 上的可行解。对每一个 h 满足 $1\leqslant h\leqslant m$，将 S_h^* 划分为 k 段子调度 $S_{h_1}^*, S_{h_2}^*, \cdots, S_{h_k}^*$，其中调度 $S_{h_1}^*$ 为从 S_h^* 中的第一个工作开始直到最后一次在 S_h^* 中出现截止期为 T_1 的工作（包括此工作）为止的工作序列，而每一调度 $S_{h_i}^*$ $(2\leqslant i\leqslant k)$ 表示从 $S_{h(i-1)}^*$ 后面的第一个工作开始直到最后一次在 S_h^* 中出现截止期为 T_i 的工作（包括此工作）为止的工作序列。如果 S_h^* 中不存在截止期为 T_i 的工作，则 $S_{h_i}^*$ 为空集。因此 $S_h^*=\langle S_{h_1}^*\diamond S_{h_2}^*\diamond\cdots\diamond S_{h_k}^*\rangle$，$1\leqslant h\leqslant m$。

定义 $P_i^*=\sum_{h=1}^{m}P(S_{h_i}^*)$ 为 S^* 中第 i 段收益 $(1\leqslant i\leqslant k)$。显然 $opt=\sum_{i=1}^{k}P_i^*$。因此有：

$$k\cdot\max\{P_1^*, P_2^*, \cdots, P_k^*\}\geqslant opt \tag{1}$$

令 $\Psi_i^*=\langle\Psi_{1i}^*, \Psi_{2i}^*, \cdots, \Psi_{mi}^*\rangle$ 表示在 m 个机器上使用工作集 $\{G_i\cup G_{i+1}\cup ?\cup G_k\}$ 在时间限制 T_i 内所获取得的一个最优调度，其中 $1\leqslant i\leqslant k$。

引理 1 对于任意一个 $1\leqslant i\leqslant k$，$P(\Psi_i^*)\geqslant P_i^*$ 成立。

证明：P_i^* 代表 S^* 中的每个机器上第 i 段子调度的收益之和。在任意一个机器 M_h $(1\leqslant h\leqslant m)$ 中，根据子调度 $S_{h_i}^*$ 的定义可知，$S_{h_i}^*$ 不会包含截止期小于 T_i 的工作，即 $S_{h_i}^*$ 中的工作都来自于集合 $\{G_i\cup G_{i+1}\cup\cdots\cup G_k\}$。因此 $\{S_{1i}^*\cup S_{2i}^*\cup\cdots\cup S_{mi}^*\}$ 是工作集 $\{G_i\cup G_{i+1}\cup\cdots\cup G_k\}$ 的子集。因为 $S_{h_i}^*$ 中的最后一个工作的截止期为 T_i，$S_{h_i}^*$ 中所有工作在机器 M_h 上的完成时刻不大于 T_i。因此，集合 $\{S_{1i}^*\cup S_{2i}^*\cup\cdots\cup S_{mi}^*\}$ 中所有工作在 m 个机器上的完成时刻都不大于 T_i。由于 Ψ_i^* 是在 m 个机器上使用工作集 $\{G_i\cup G_{i+1}\cup\cdots\cup G_k\}$ 在时间限制为 T_i 内选取的一个最优调度，Ψ_i^* 的收益不小于 $\{S_{1i}^*\cup S_{2i}^*\cup\cdots\cup S_{mi}^*\}$ 的收益，故 $P(\Psi_i^*)\geqslant P_i^*$ 成立。引理 1 得证。

4.2 MFPD_k 算法设计

本节中将针对 $\mathrm{MFPD}_k(G,D_k,m)$ 问题设计一个近似算法。

定理 1 如果 MFPD_1 问题存在一个近似比为 α 的多项式时间近似算法 A_1，那么 MFPD_k 问题就存在一个近似比为 $k\alpha$ 的多项式时间近似算法 A_k。

证明：考虑 MFPD_k 问题的以下近似算法。

算法 1 A_k 算法

输入：MFPD_k 问题实例 (G,D_k,m)，其中 $D_k=\{T_1,T_2,\cdots,T_k\}$，$T_1<$

$T_2 < \cdots < T_k$

输出：工作集 G 在 m-机器上的一个可行子集G′以及子集G′在 m-机器上的一个可行解$S'=\langle S_1', S_2', \cdots, S_m' \rangle$

1. 将工作集 G 按截止期划分为 k 个子集$G_1, G_2, \cdots, G_k$，使得G_i中的工作的截止期是T_i，其中$1 \leqslant i \leqslant k$

2. $S'=\langle\ \rangle$

3. for i=1 to k do

3.1. 对 $MFPD_1$问题的实例$(G_i \cup G_{i+1} \cup \cdots \cup G_k, \{T_i\}, m)$使用算法$A_1$，得到解$S_{Ai}=\langle S_{1i}, S_{2i}, \cdots, S_{mi} \rangle$

3.2. if $P(S_{Ai}) > P(S')$ then $S'=S_{Ai}$

4. 输出S′

首先证明A_k算法的输出S′是 $MFPD_k$问题实例(G, D_k, m)的一个可行解。工作序列集合S_{Ai}是 $MFPD_1$问题的实例$(G_i \cup G_{i+1} \cup \cdots \cup G_k, \{T_i\}, m)$的可行解。$S_{Ai}$中的工作均来自于$\{G_i \cup G_{i+1} \cup \cdots \cup G_k\}$，所以在$S_{Ai}$中截止期最小的工作的截止期不小于$T_i$。又因为调度$S_{Ai}$中的工作均可在时刻$T_i$前完成，所以$S_{Ai}$中的工作都能在不超过各自的截止期前完成。现在将调度S_{Ai}放入 $MFPD_k$问题实例(G, D_k, m)中的 m 个机器上，调度S_{Ai}中的每一个工作仍然能在不超过各自的截止期前完成，因此S_{Ai}是 $MFPD_k$问题实例(G, D_k, m)的可行解。由于算法的输出S′选择的是收益最大的S_{Ai}，故S′是 $MFPD_k$问题实例(G, D_k, m)的一个可行解。

接下来证明该算法的近似比为 $k\alpha$，即证明 $k\alpha \cdot P(S') \geqslant opt$。根据算法$A_1$解的性质并结合引理 1，可知：

$$\begin{aligned} k\alpha \cdot P(S') &\geqslant \alpha \cdot \sum_{i=1}^{k} P(S_{Ai}) \\ &\geqslant \sum_{i=1}^{k} P(\Psi_i^*) \\ &\geqslant \sum_{i=1}^{k} P_i^* = opt \end{aligned} \tag{2}$$

其中，第一个不等式是因为 $P(S')$等于最大的 $P(S_{Ai})$，第二个不等式是因为算法A_1的近似比不大于 α，而Ψ_i^*是 m 个机器上使用工作集$\{G_i \cup G_{i+1} \cup \cdots \cup G_k\}$在时间限制$T_i$内所获取得的一个最优调度，第三个不等式来自于引理 1。

式(2)表明算法 1 的解不小于$1/(k\alpha)$倍的最优解，即算法A_k的近似比为 $k\alpha$，其中 k 是截止期的个数。

下面分析算法 1 的时间复杂度。算法 1 中的步骤 1 将工作集 G 分为了 k 个工作子集，时间复杂度为 $O(n)$。步骤 2 是一个初始化的过程，时间复杂度为 $O(1)$。步骤 3 循环 k 次，每次调用近似算法A_1，故步骤 3 花费的时间是算法A_1的 k 倍。因此，如果算法A_1的运行时间为多项式，则算法 1 即算法A_k的运行时间也是多项式。

综上所述，算法A_k是 $MFPD_k$问题的一个近似比为 $k\alpha$ 的多项式时间近似算法。定理 1 得证。

近年来，$MFPD_1$问题引起调度领域的注意和兴趣[21-22]。结合定理 1 和 $MFPD_1$问题近似算法近年来的最新结果，我们可得到以下推论(其中$\epsilon > 0$是任意固定常数)：

推论 1 1)$MFPD_k$问题存在一个近似比为$(3k+\epsilon)$的多项式时间近似算法；2)当机器个数 m 为固定常数时，$MFPD_k$问题存在一个近似比为$(k+\epsilon)$的多项式时间近似算法。

证明：推论 1)：Chen 等[21]结合多重背包问题近似算法研究的最新进展，对 $MFPD_1$问题提出了一个近似比为$(3+\epsilon)$的多项式时间近似算法。将此算法结合定理 1，即得到 $MFPD_k$问题的一个近似比为$(3k+\epsilon)$的多项式时间近似算法。

推论 2)：对机器个数 m 为固定常数的情况，Tong 等[22]对 $MFPD_1$问题提出了一个多项式时间近似方案，即近似比为$(1+\epsilon)$的多项式时间近似算法。将此算法结合定理 1，即得到 $MFPD_k$问题的一个近似比为$(k+\epsilon)$的多项式时间近似算法。

推论 1 得证。

由推论 1 直接推出：当截止期个数 k 为固定常数时，$MFPD_k$问题存在一个常数近似比的多项式时间近似算法。

5 $MFPD_{k,R \leqslant T}$模型算法设计

本章考虑在单个两阶段机器上 $MFPD_{k,R \leqslant T}$模型的调度问题，即 $MFPD_{k,R \leqslant T}(G, D_k, 1)$调度问题。在此模型下，工作集 G 中的任意一个工作 J 的 R-处理时间不大于 T-处理时间。

5.1 理论准备

首先证明以下引理：

引理 2 令$S'=\langle J_1', J_2', \cdots, J_q' \rangle$为 $MFPD_{k,R \leqslant T}(G, D_k, 1)$模型下单个两阶段机器上按 T-处理时间非递增排序而成的序列(即调度)，则在S′中所有工作的 T-处理是无间隙连续执行的。

证明：假设对任一 x，工作J_x'的 R-阶段处理时间与 T-阶段处理时间分别为r_x和t_x。对于S′ 中任意两个相邻的工作J_x'，J_{x+1}'，由于每个工作的 R-处理时间不大于 T-处理时间，因此有：

$$t_x \geqslant r_x, t_{x+1} \geqslant r_{x+1} \tag{3}$$

又因为S′中的工作是按照 T-处理时间非递增排序，因此有：

$$t_x \geqslant t_{x+1} \tag{4}$$

结合式(3)和式(4)可知：

$$t_x \geqslant r_{x+1} \tag{5}$$

根据两阶段工作在两阶段机器上执行的规则，工作J_{x+1}'的 T-处理不仅要等待自己的 R-处理结束，而且还要等待J_x'的 T-处理结束后才能开始执行。由于在 S′中工作J_x'，J_{x+1}'的 R-处理是连续执行的，所以J_x'的 T-处理开始时刻不会比J_{x+1}'的 R-处理开始时刻更早。式(5)表明，J_x'的 T-处理时间不会比J_{x+1}'的 R-处理时间短，且J_x'的 T-处理结束时刻不早于J_{x+1}'的 R-处理结束时刻，因此J_{x+1}'的 T-处理只需等待J_x'的 T-处理完成后即可立即执行。也就是说，J_x'的 T-处理与J_{x+1}'的 T-处理之间没有间隙。由于J_x'和J_{x+1}'是S′中任意两个连续的工作，这证明了在调度S′中所有工作的 T-处理是无间隙连续执行的。引理 2 得证。

在以下讨论中，假设S^*为 $MFPD_{k,R \leqslant T}$实例$(G, D_k, 1)$在单个两阶段机器上工作集合 G 的一个最优调度。对每一个$i(1 \leqslant i \leqslant k)$令$S_i^* = \langle J_{i1}^*, J_{i2}^*, \cdots, J_{i\#}^* \rangle$表示由$S^*$中截止期为$T_i$的工作按照 T-处理时间非递增排序而成的序列(即调度)。如果在S^*中不存在截止期为T_i的工作，则令$S_i^* = \emptyset$。注意，因为两个调度S_i^*和S_j^*中的工作数量可能不同，这里 # 并不是一个固定数字，只是用$J_{i\#}^*$来表示S_i^*中的最后一个工作。

定理 2 对于每一 $1\leqslant i\leqslant k$，令 $S_i^* \setminus J_{i1}^*$ 为删除 S_i^* 中的第一个工作 J_{i1}^* 并仍保持其余工作的先后顺序所得到的序列。则在拼接调度 $S_+^{*\prime\prime}=(S_1^* \setminus J_{11}^*)+(S_2^* \setminus J_{21}^*)+\cdots+(S_k^* \setminus J_{k1}^*)$ 中，每一个工作都能在不晚于其截止期前完成，即拼接调度 $S_+^{*\prime\prime}$ 是一个可行调度。

证明：将构造拼接调度 $S_+^{*\prime\prime}$ 的过程分为两步：

第 1 步 对于每一个 $1\leqslant i\leqslant k$，将 J_{i1}^* 的 R-处理时间 r_{i1} 置为 0，而将工作 J_{i1}^* 变为工作 $J_{i1}^{*\prime}$。然后构造一个新的调度 $S_i^{*\prime}=\langle J_{i1}^{*\prime}, J_{i2}^*, \cdots, J_{i\#}^* \rangle$。由于 $J_{i1}^{*\prime}$ 的 R-处理时间为 0，调度 $S_i^{*\prime}$ 中 $J_{i1}^{*\prime}$ 的 T-处理开始时刻和 $J_{i2}^{*\prime}$ 的 R-处理开始时刻均为 0。另外，注意调度 $S_i^{*\prime}$ 中的工作仍然满足引理 2 的条件。由引理 2 可知，调度 $S_i^{*\prime}$ 中所有工作的 T-处理是无间隙连续执行的。

将子调度 $S_1^{*\prime}, S_2^{*\prime}, \cdots, S_k^{*\prime}$ 拼接成调度 $S_+^{*\prime\prime}=\langle S_1^{*\prime}+S_2^{*\prime}+\cdots+S_k^{*\prime}\rangle$。由于对每个 $1\leqslant i\leqslant k$，调度 $S_i^{*\prime}$ 中的第一个工作 $J_{i1}^{*\prime}$ 的 T-处理开始时刻为 0，在拼接调度 $S_+^{*\prime\prime}$ 中，对每个 $1\leqslant i\leqslant k-1$，调度 $S_{i+1}^{*\prime}$ 中第一个工作 $J_{(i+1)1}^{*\prime}$ 的 T-处理开始时刻与调度 $S_i^{*\prime}$ 中最后一个工作 $J_{i\#}^*$ 的 T-处理结束时刻相同，即调度 $S_i^{*\prime}$，$S_{i+1}^{*\prime}$ 之间的 T-处理没有间隙。因此，拼接调度 $S_+^{*\prime\prime}$ 中第一个工作的 T-处理开始时刻为 0，且所有工作的 T-处理是无间隙连续执行的。

我们证明拼接调度 $S_+^{*\prime\prime}$ 中所有工作都能在不晚于其截止期完成。对每个 $1\leqslant i\leqslant k$，定义 $M_i^{*\prime}$ 为在拼接调度 $S_+^{*\prime\prime}$ 中截止期为 T_i 的最后一个工作 $J_{i\#}^*$ 的完成时刻。由于拼接调度 $S_+^{*\prime\prime}$ 中所有工作的 T-处理是无间隙连续执行的，$M_i^{*\prime}=\sum_{x=1}^{i}\sum_{j=1}^{\#}t_{xj}$，这里我们假设对所有 x 和 j，工作 J_{xj}^* 的 T-处理时间是 t_{xj}。定义 $M_{i\text{-}last}^*$ 为在最优调度 S^* 中最后一个截止期不大于 T_i 的工作 $J_{i\text{-}last}^*$ 的完成时刻并假设 $J_{i\text{-}last}^*$ 的截止期为 $T_{left\text{-}i}$，则 $T_{left\text{-}i}\leqslant T_i$。由于 $J_{i\text{-}last}^*$ 是最后一项截止期不大于 T_i 的工作，其完成时刻 $M_{i\text{-}last}^*$ 一定不小于所有在最优调度 S^* 中截止期不大于 T_i 的工作的 T-处理时间之和。结合不等式 $T_{left\_i}\leqslant T_i$ 则有：

$$M_i^{*\prime}\leqslant M_{i\text{-}last}^*\leqslant T_{left\text{-}i}\leqslant T_i \tag{6}$$

式(6)表明，在拼接调度 $S_+^{*\prime\prime}$ 中，每一个截止期为 T_i 的工作的完成时刻都不大于 T_i，即截止期为 T_i 的工作都能在不晚于时间 T_i 前完成。因这一结论对所有 $1\leqslant i\leqslant k$ 都成立，所以拼接调度 $S_+^{*\prime\prime}$ 中的所有工作都能在不晚于其截止期完成。

第 2 步 在调度 $S_+^{*\prime\prime}$ 中，对每一个 $1\leqslant i\leqslant k$，删掉 $S_i^{*\prime}$ 中的工作 $J_{i1}^{*\prime}$ 而不改变其他工作的开始和完成时刻。

我们先证明第 2 步得到的工作集合 $(S_1^* \setminus J_{11}^*)\cup\cdots\cup(S_k^* \setminus J_{k1}^*)$ 是一个有效调度 S''。工作 $J_{i1}^{*\prime}$ 的 R-处理时间为 0，故删掉 $J^{*\prime}{}_{i1}$ 不需改变其他工作在调度 $S^{*\prime\prime}{}_+$ 中的 R-处理开始和结束时刻，故剩余工作在调度 $S_+^{*\prime\prime}$ 中仍保持 T-处理开始时刻不早于其 R-处理结束时刻的性质，即 S'' 是一个有效调度。同时，由于拼接调度 $S_+^{*\prime\prime}$ 中的所有工作都能在不晚于其截止期完成而第 2 步的操作不改变任何工作的结束时刻，调度 S'' 中的所有工作都能在不晚于其截止期完成。

最后，在不改变调度中工作顺序以及工作 R-处理开始时刻的前提下，将调度 S'' 中每个工作的 T-处理开始时刻尽量往前移且保持调度的有效性(即令每一工作的 T-处理开始时刻等于前一工作的 T-处理结束时刻和本身的 R-处理结束时刻中较大的值)。很容易验证，这一操作所得的调度就是拼接调度 $S_+^{*\prime\prime}=(S_1^* \setminus J_{11}^*)+(S_2^* \setminus J_{21}^*)+\cdots+(S_k^* \setminus J_{k1}^*)$。由于这一在调度 S'' 上的操作不会延迟任一工作的结束时刻，且调度 S'' 中的所有工作都能在不晚于其截止期完成，我们得到结论：在拼接调度 $S_+^{*\prime\prime}=(S_1^* \setminus J_{11}^*)+(S_2^* \setminus J_{21}^*)+\cdots+(S_k^* \setminus J_{k1}^*)$ 中，每一个工作都能在不晚于其截止期完成。定理 2 得证。

5.2 最大收益拼接调度算法设计

如定理 2 所证，将工作按截止期分组调度然后拼接得到的拼接调度 $S_+^{*\prime\prime}=(S_1^* \setminus J_{11}^*)+\cdots+(S_k^* \setminus J_{k1}^*)$ 是 $\mathrm{MFPD}_{k,R\leqslant T}$ 模型中实例 $(G,D_k,1)$ 的一个可行调度。由于拼接调度 $S_+^{*\prime\prime}$ 是由实例 $(G,D_k,1)$ 的最优解 S^* 中对每个截止期删掉一个工作得到的，$S_+^{*\prime\prime}$ 也是实例 $(G,D_k,1)$ 的最优解的一个不错的近似。本节讨论如何构造一个最大收益的按截止期分组调度构造的拼接调度。

对于一个两阶段机器上调度只有一个公共截止期 T 的两阶段工作集 G，即 $\mathrm{MFPD}_1(G,\{T\},1)$ 问题，Dawande 等[20]提出了一个动态规划算法可以在 $O(n\cdot T^2)$ 时间内找到最优解。我们将该动态规划算法命名为 $\mathrm{Dyn}(G,T)$。考虑以下算法。

算法 2 算法 BestSplice$(G,D_k,1)$

输入：$\mathrm{MFPD_k}$ 问题实例 $(G,D_k,1)$，其中 $D_k=\{T_1,T_2,\cdots,T_k\}$

输出：实例 $(G,D_k,1)$ 的一个可行解

1. for $(1\leqslant i\leqslant k)$ 令 G_i 为 G 中截止期为 T_i 的工作的集合
2. $S=\langle\ \rangle$
3. for$(0\leqslant D_1\leqslant T_1; D_1\leqslant D_2\leqslant T_2;\cdots; D_{k-1}\leqslant D_k\leqslant T_k)$

$S_1=\mathrm{Dyn}(G_1,D_1)$

$S_2=\mathrm{Dyn}(G_2,D_2-D_1)$

…

$S_k=\mathrm{Dyn}(G_k,D_k-D_{k-1})$

4. if $(P(S_1)+P(S_2)+\cdots+P(S_k)>P(S))$ $S=\langle S_1\diamond S_2\diamond\cdots\diamond S_k\rangle$
5. 输出(S)

定理 3 算法 BestSplice 输出的调度 S 是实例 $(G,D_k,1)$ 的一个可行调度，其收益不小于任何实例 $(G,D_k,1)$ 的按截止期分组拼接得到的可行拼接调度。

证明：考虑拼接调度 $S'=\langle S_1+S_2+\cdots+S_k\rangle$。第 3 步中第 1 条语句 $S_1=Dyn(G_1,D_1)$ 保证在拼接调度 S' 下序列 S_1 中的所有工作(其截止期均为 T_1)都可在时刻 $D_1\leqslant T_1$ 前完成。现归纳假设对 i，$1\leqslant i\leqslant k-1$，在拼接调度 S' 下序列 S_i 中的所有工作(其截止期均为 T_i)都可在时刻 $D_i\leqslant T_i$ 前完成。由于第 3 步的第 $i+1$ 条语句 $S_{i+1}=Dyn(G_{i+1},D_{i+1}-D_i)$ 保证了在调度 S_{i+1} 下，序列 S_{i+1} 中所有工作均可在时间限制 $D_{i+1}-D_i$ 内完成，结合归纳假设，我们得到在拼接调度 $S'=\langle S_1+\cdots+S_i+S_{i+1}+\cdots+S_k\rangle$ 中，序列 S_{i+1} 中所有工作均可在时刻 $D_i+D_{i+1}-D_i=D_{i+1}\leqslant T_{i+1}$ 前完成。这就完成了归纳证明，从而证明了拼接调度 S' 为可行调度。

连接调度 $S=\langle S_1\diamond S_2\diamond\cdots\diamond S_k\rangle$ 可通过将拼接调度 $S'=\langle S_1+S_2+\cdots+S_k\rangle$ 的工作的开始时刻尽量往前移得到。为此，从调度 S' 的第 1 个工作开始，依次将每个工作的 R-处理开始时刻置为前一工作的 R-处理结束时刻，然后将其 T-处理开始时刻置为前一工作的 T-处理结束时刻和本工作的 R-处

理结束时刻中较大的值。这一操作不会延迟任何工作的结束时刻，因此第4步所得连接调度，以及算法输出的调度 S，都是一个可行调度。

下面证明算法输出的连接调度 $S=\langle S_1 \diamond S_2 \diamond \cdots \diamond S_k \rangle$ 的收益不小于任何按截止期分组得到的可行拼接调度的收益。因为连接调度 S 的收益等于其对应的拼接调度 $S'=\langle S_1+S_2+\cdots+S_k \rangle$ 的收益，我们只需证明拼接调度 S' 的收益不小于任何按截止期分组得到的可行拼接调度的收益。令 $S^+=\langle S_1^+ + S_2^+ + \cdots + S_k^+ \rangle$ 为任一按截止期分组得到的可行拼接调度，其中对每一个 i，S_i^+ 是由截止期为 T_i 的工作组成的序列。令 $D_0^+=0$，对 $i>0$，令 D_i^+ 为拼接调度 S^+ 中序列 S_i^+ 最后一个工作的结束时刻。由于 S^+ 是可行调度，$D_i^+ \leqslant T_i$。按拼接调度的定义，每个序列 S_i^+ 的开始时刻为 D_{i-1}^+，结束时刻为 D_i^+。从而，如果将 S_i^+ 看作一个独立的调度，则 S_i^+ 的完工时刻为 $D_i^+ - D_{i-1}^+$，即 S_i^+ 是实例 $\mathrm{MFPD}_1(G_i, \{D_i^+ - D_{i-1}^+\}, 1)$ 的一个可行解。当算法 BestSplice 第3步对所有 i 取值 $D_i=D_i^+$ 时，我们得到 $S_i=Dyn(G_i, D_i - D_{i-1})$ 是实例 $\mathrm{MFPD}_1(G_i, \{D_i^+ - D_{i-1}^+\}, 1)$ 的一个最优解。因此，$P(S_i) \geqslant P(S_i^+)$。由于这个不等式对所有 i 都成立，当第3步取这些 D_i 值时，将得到一个收益不小于 $P(S^+)$ 的可行调度。由于算法第4步保证算法的输出调度 S 是第3步所得可行调度中收益最大的调度，算法输出的调度 S 的收益 $P(S)$ 不会小于 $P(S^+)$。因为 S^+ 是实例 $(G, D_k, 1)$ 的任意按截止期分组得到的可行拼接调度，定理3得证。

下面分析算法 BestSplice 的时间复杂度，主要是算法第3步的时间复杂度。由于对所有 i，$D_i \leqslant T_i$，算法第3步循环的次数不大于 $T_1 \times T_2 \times \cdots \times T_k \leqslant T_k^k$。每次循环调用 k 次算法 Dyn，共花时间 $O(kn \cdot T_k^2)$。因此，算法 BestSplice 的时间复杂度为 $O(kn \cdot T_k^{k+2})$。

5.3 $\mathrm{MFPD}_{k,R\leqslant T}$ 算法设计

有了以上准备，我们得到以下 $\mathrm{MFPD}_{k,R\leqslant T}$ 问题的算法。

算法3 算法 $\mathrm{MFPD}_{k,R\leqslant T}(G, D_k, 1)$

输入：$\mathrm{MFPD}_{k,R\leqslant T}$ 问题实例 $(G, D_k, 1)$，其中 $D_k=\{T_1, T_2, \cdots, T_k\}$

输出：实例 $(G, D_k, 1)$ 的一个可行解

1. $S=\langle\ \rangle$
2. for ($1 \leqslant i \leqslant k$) 令 G_i 为 G 中截止期为 T_i 的工作的集合
3. for (each J_i^* in G_i, $1 \leqslant i \leqslant k$)
   3.1. if ($\{J_1^*, J_2^*, \cdots, J_k^*\}$ 是1-机器上的可行集合)
   构造 $\{J_1^*, J_2^*, \cdots, J_k^*\}$ 的一个可行调度 S_1
   3.2. if ($P(S_1) > P(S)$) $S=S_1$
   3.3. for ($1 \leqslant i \leqslant k$)
   $G_i'=G_i$ 中 T-处理时间不大于 J_i^* 的所有工作
   3.4. $G'=G_1' \cup G_2' \cup \cdots \cup G_k'$
   3.5. $S_2=\mathrm{BestSplice}(G', D_k, 1)$
   3.6. if ($P(S_2) > P(S)$) $S=S_2$
4. 输出(S)

定理4 算法 $\mathrm{MFPD}_{k,R\leqslant T}$ 的近似比为2。

证明：采用5.1节中的符号。令调度 S^* 为 $\mathrm{MFPD}_{k,R\leqslant T}$ 实例 $(G, D_k, 1)$ 的一个最优调度。对每一个 i，令 $S_i^*=\langle J_{i1}^*, J_{i2}^*, \cdots, J_{i\#}^* \rangle$ 为 S^* 中截止期为 T_i 的工作按照 T-处理时间非递增排序而成的序列。当算法 $\mathrm{MFPD}_{k,R\leqslant T}$ 的第3步对所有 $1 \leqslant i \leqslant k$ 考虑工作 $J_i^*=J_{i1}^*$ 时，$\{J_{11}^*, J_{21}^*, \cdots, J_{k1}^*\}$ 显然是一个可行集合（因为它是可行调度 S^* 的一个子集）。因此，算法3第3.2步保证了算法输出的调度的收益不小于 $P(\{J_{11}^*, J_{21}^*, \cdots, J_{k1}^*\})$。

按照算法3第3.4步所给的工作集合 G' 的定义，按截止期分组拼接所得的拼接调度 $S_+^{*\prime\prime}=(S_1^* \setminus J_{11}^*)+(S_2^* \setminus J_{21}^*)+\cdots+(S_k^* \setminus J_{k1}^*)$ 是 $\mathrm{MFPD}_{k,R\leqslant T}$ 问题实例 $(G', D_k, 1)$ 的一个调度。由定理2，拼接调度 $S_+^{*\prime\prime}$ 是一个可行调度。由定理3，由算法3第3.5步得到的调度 S_2 是 $\mathrm{MFPD}_{k,R\leqslant T}$ 问题实例 $(G', D_k, 1)$ 的一个可行调度，其收益不小于任何按截止期分组的拼接调度。因此第3.6步保证了算法输出的调度的收益不小于 $P(S_+^{*\prime\prime})$。

由于实例 $(G', D_k, 1)$ 最优解值 $P(S^*)$ 等于 $P(\{J_{11}^*, J_{21}^*, \cdots, J_{k1}^*\})+P(S_+^{*\prime\prime})$，我们得到结论，算法输出的调度的收益不小于最优解值的1/2，即算法 $\mathrm{MFPD}_{k,R\leqslant T}$ 的近似比为2。定理4得证。

分析算法 $\mathrm{MFPD}_{k,R\leqslant T}$ 的时间复杂度，其主要步骤是第3步。每个子集 G_i 包含不超过 n 个工作。第3步是遍历一个集合，该集合中的每个元素由 $G_1, G_2, \cdots, G_k$ 中各取一个元素组成。因此，第3步的循环一共执行不超过 n^k 次。第3步循环每次执行的时间主要是第3.5步调用算法 $\mathrm{BestSplice}(G', D_k, 1)$。由5.2的讨论可知，算法 BestSplice 的时间复杂度为 $O(kn \cdot T_k^{k+2})$。同时注意到第3.1步的执行时间不会超过 $O(n \cdot k!)$（即将所有 $\{J_1^*, J_2^*, \cdots, J_k^*\}$ 的排序检查一遍），得出结论：算法 $\mathrm{MFPD}_{k,R\leqslant T}$ 的时间复杂度为 $O(n^k(k! + n \cdot T_k^{k+2}))$。因此，当截止期个数 k 是固定常数时，算法 $\mathrm{MFPD}_{k,R\leqslant T}$ 是近似比为2的伪多项式时间算法。

结束语 基于云计算领域研究和应用的需求，本文建立了以追求收益最大化的云计算系统工作调度问题的理论模型 MFPD_k。现有文献中有关研究只考虑了这一模型下只有一个公共截止期的工作调度问题。本文首次研究了此模型下带有多个截止期的工作调度问题。当截止期的个数为一固定常数时，本文提出了常数近似比的多项式时间近似算法。基于云计算应用的一种特殊情况，即当磁盘读写速度不慢于网络传输速度时，本文建立了模型 $\mathrm{MFPD}_{k,R\leqslant T}$。当截止期的个数为一固定常数时，为该模型提出了一个近似比为2的伪多项式时间近似算法。

本文提出的近似算法的时间复杂度较高，算法的近似比也还不够理想。进一步的研究工作将着重于改善算法的近似比和降低算法的时间复杂度。

参考文献

[1] ARMBRUST M, FOX A, GRIFFITH R, et al. A view of cloud computing[J]. Communications of ACM, 2010, 53(4): 50-58.

[2] RITTINGHOUSE J, RANSOME J. Cloud Computing: Implementation, Management, and Security[M]. CRC press, 2017.

[3] ZHANG Y, ZHOU Y. TransOS: a transparent computing-based operating system for the cloud[J]. International Journal of Cloud Computing, 2012, 1(4): 287-301.

[4] ZHANG Y, DUAN S, ZHANG D, et al. Transparent Computing:

Development and Current Status [J]. Chinese Journal of Electronics, 2020, 29(5): 793-811.

[5] ZHANG Y, ZHOU Y. Separating computation and storage with storage virtualization [J]. Computer Communications, 2011, 34(13): 1539-1548.

[6] POTTS C, STRUSEVICH V. Fifty years of scheduling: a survey of milestones[J]. Journal of the Operational Research Society, 2009, 60(1): S41-S68.

[7] JOHNSON S. Optimal two- and three-stageproduction schedules with setup times included[J]. Naval Research Logistics Quarterly, 1954, 1(1): 61-68.

[8] GAREY M, JOHNSON D. Computers andIntractability: A Guide to the Theory of NP-Completeness. [M]. San Francisco: Freeman, 1979.

[9] KOVALYOV M. Efficient epsilon-approximation algorithm for minimizing the makespan in a parallel two-stage system [J]. Vesti Academii navuk Belaruskai SSR, Ser. Phiz. -Mat. Navuk, 1985, 3: 119.

[10] DONG J, TONG W, LUO T, et al. An FPTAS for the parallel two-stage flowshop problem[J]. Theoretical Computer Science, 2017, 657: 64-72.

[11] WU G, CHEN, J, WANG J. Scheduling two-stage jobs on multiple flowshops[J]. Theoretical Computer Science, 2019, 776: 117-124.

[12] WU G, CHEN J, WANG J. On scheduling multiple two-stage flowshops[J]. Theoretical Computer Science, 2020, 818: 74-82.

[13] WU G, CHEN J, WANG J. On scheduling inclined jobs on multiple two-stage flowshops [J]. Theoretical Computer Science, 2019, 786: 67-77.

[14] WU G, CHEN J, WANG J. Improved approximation algorithms for two-stage flowshops scheduling problem [J]. Theoretical Computer Science, 2020, 806: 509-515.

[15] DONG J, JIN R, LUO T, et al. A polynomial-time approximation scheme for an arbitrary number of parallel two-stage flow-shops [J]. European Journal of Operational Research, 2020, 281(1): 16-24.

[16] CONWAY R, MAXWELL W, MILLER L. Theory of scheduling [M]. Courier Corporation, 2003.

[17] CHEKURI C, KHANNA S. A polynomial time approximation scheme for the multiple knapsack problem[J]. SIAM Journal on Computing, 2005, 35(3): 713-728.

[18] CHEN L, ZHANG G. Packing groups of items into multiple knapsacks[J]. ACM Transactions on Algorithms (TALG), 2018, 14(4): 1-24.

[19] CROCE F, GUPTA J, TADEI R. Minimizing tardy jobs in a flowshop with common due date[J]. European Journal of Operational Research, 2000, 120(2): 375-381.

[20] DAWANDE M, GAVIRNENI S, RACHAMADUGU R. Scheduling a two-stage flowshop under makespan constraint [J]. Mathematical and Computer Modelling, 2006, 44(1/2): 73-84.

[21] CHEN J, HUANG M, GUO Y. Scheduling multipletwo-Stage flowshops with a deadline[J]. Theoretical Computer Science, 2022, 921: 100-111.

[22] TONG W, XU Y, ZHANG H. A polynomial-time approximation scheme for parallel two-stage flowshops under makespan constraint[J]. Theoretical Computer Science, 2022, 922: 438-446.

LIU Zhimin, born in 1998, postgraduate. His main research interests include computer optimization algorithms and applications.

CHEN Jianer, born in 1954, professor. His main research interests include algorithms and their applications, network optimization, and computer graphics.