软件学报 ISSN 1000-9825, CODEN RUXUEW
2024,35(11):5319−5340 [doi: 10.13328/j.cnki.jos.007037] [CSTR: 32375.14.jos.007037]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向设备与人力资源协作任务的优化调度方法*

何飞佳[1,2], 沈立炜[1,2], 宋育臣[1,2], 牛军钰[1,2], 赵文耘[1,2]

[1](复旦大学 计算机科学技术学院, 上海 200438)

[2](上海市数据科学重点实验室 (复旦大学), 上海 201203)

通信作者: 沈立炜, E-mail: shenliwei@fudan.edu.cn

摘　要: 面向人机物融合的泛在计算正成为软件发展的新需求和新趋势, 基于这种新形态计算模式的人机物融合应用将软件技术进一步拓展至对线下资源, 包括物理设备和人力资源的有效利用. 作为典型的人机物融合场景, 物理世界中设备资源与人力资源间的协作具有资源可选性、任务高频性、工人动态性的特点, 传统的资源调度技术无法有效应对该类型任务 (简称为 DHRC 任务) 中的调度需求. 为此, 提出一种面向设备与人力资源协作任务的优化调度方法, 所提方法分为设备资源调度和人力资源调度两个阶段. 在设备资源调度阶段, 提出基于 NSGA-II 的设备资源调度算法, 在综合考虑任务距离、设备负载和设备位置周边工人人数等因素情况下实现任务对资源的优化选择. 在人力资源调度阶段, 提出基于 DPSO 的人力资源调度算法, 根据工人位置和协作依赖等因素实现对工人的优化选择以及相应的路径规划. 在模拟环境内的实验结果表明, 所提方法第 1 阶段的算法在效率上与对比算法相当, 在效用性上优于对比算法 (离散粒子群优化算法). 第 2 阶段的算法在效率上与效用性上均优于对比算法 (使用锦标赛机制改进的遗传算法).

关键词: 人机物; 资源调度; NSGA-II; DPSO

中图法分类号: TP316

中文引用格式: 何飞佳, 沈立炜, 宋育臣, 牛军钰, 赵文耘. 面向设备与人力资源协作任务的优化调度方法. 软件学报, 2024, 35(11): 5319–5340. http://www.jos.org.cn/1000-9825/7037.htm

英文引用格式: He FJ, Shen LW, Song YC, Niu JY, Zhao WY. Optimal Scheduling Approach for Device and Human Resource Collaborative Tasks. Ruan Jian Xue Bao/Journal of Software, 2024, 35(11): 5319–5340 (in Chinese). http://www.jos.org.cn/1000-9825/7037.htm

Optimal Scheduling Approach for Device and Human Resource Collaborative Tasks

HE Fei-Jia[1,2], SHEN Li-Wei[1,2], SONG Yu-Chen[1,2], NIU Jun-Yu[1,2], ZHAO Wen-Yun[1,2]

[1](School of Computer Science, Fudan University, Shanghai 200438, China)

[2](Shanghai Key Laboratory of Data Science (Fudan University), Shanghai 201203, China)

Abstract: Ubiquitous computing for human-cyber-physical integration is becoming a new requirement and trend in software development. Based on this new computing paradigm, human-cyber-physical applications further extend software technology to the effective utilization of offline resources, including physical devices and human resources. As a typical human-cyber-physical scenario, the collaboration between the device and human resources in the physical world features resource selectivity, high task frequency, and worker dynamics. Traditional resource scheduling techniques cannot meet the scheduling requirements of this task type (referred to as DHRC task). Thus, this study proposes an optimal scheduling method for collaborative tasks between device and human resources. This method includes two stages of device resource scheduling and human resource scheduling. In the device resource scheduling stage, a device resource scheduling algorithm based on NSGA-II is proposed to optimize task resource selection by comprehensively considering such factors as task distance, device load, and the worker number around the device location. In the human resource scheduling stage, a human resource scheduling algorithm based on DPSO is put forward to optimize the worker selection and corresponding path planning according to such factors as worker location and collaboration dependency. Experiments in a simulated environment show that the algorithm in the first stage is

* 收稿时间: 2023-03-27; 修改时间: 2023-05-16, 2023-07-20; 采用时间: 2023-08-18; jos 在线出版时间: 2024-01-10
CNKI 网络首发时间: 2024-01-12

equivalent in efficiency and superior in utility to the compared algorithm (discrete particle swarm optimization algorithm). The algorithm in the second stage is superior in efficiency and utility to the compared algorithm (the genetic algorithm improved by the tournament mechanism).

Key words: human-cyber-physical; resource scheduling; NSGA-II; DPSO

面向人机物融合的泛在计算正在成为软件发展的新需求和新趋势[1]. 在这种新形态的计算模式下, 来自社会空间、信息空间和物理空间的各类异质异构资源通过合理的编排和协作实现用户的需求[2]. 不同于传统的由 Web Service 服务编排构成的应用, 面向人机物融合的应用突破了虚实边界, 将软件技术进一步拓展至对线下资源 (包括物理设备和人力资源) 的有效利用.

物理世界中的设备资源与人力资源之间的协作是典型的人机物融合场景. 其中, 物理设备接收指令按需地执行并生产出物理对象, 例如打印机打印出纸张、咖啡机烧制出咖啡等; 人力资源 (即志愿者, 本文称其为工人) 则可承担对这些生产出的物理对象的递送. 通过软件定义技术, 这些资源所提供的服务被封装为软件接口并被集成在人机物融合系统或平台中, 以融合应用的形式实现对这些资源服务的按需调用[2]. 例如, 用户在忙碌时请求烧制一杯咖啡, 烧制完成后请志愿者前往咖啡机所处位置拿取咖啡并送至用户所在位置. 我们将属于这种融合应用的实例称为“设备与人力资源协作任务”, 简称为 DHRC 任务 (device and human resource collaboration task).

现实世界中的 DHRC 任务具有资源可选性、任务高频性、工人动态性的特点. 资源可选性表示一个任务中的资源并非被预先指定, 而可以根据对资源的需求在运行时动态确定, 即在一组可用资源中根据实际条件进行选择. 例如, 若有多台咖啡机能够提供烧咖啡服务, 在一个特定任务执行中选择哪一台咖啡机可以根据位置距离、机器状态等因素进行选择. 任务高频性表示相同或不同的任务可能大量地同时发生. 例如, 在 1 min 内不同用户发起了多次咖啡服务. 工人动态性则表示能够承担递送任务的志愿者的位置可能发生持续变化. 这些特点使得人机物融合系统需要提供针对资源的有效调度方案, 从而应对资源的竞争需求并提高资源的利用效率.

资源调度是规划领域的重要研究方向. 传统的资源调度技术面向信息空间, 旨在提高计算、存储、通信等资源的利用率. Hadoop YARN[3], Fuxi[4], Mesos[5] 等成熟的资源调度解决方案已被广泛应用于云计算领域. 当前, 也有一些研究人员关注线下车间物理设备的调度. Zhang 等人[6] 提出了一种基于博弈论的灵活车间调度动态优化模型, 根据其实时状态将任务分配给最优机器设备. Yuan 等人[7] 提出了可重构装配线的多目标优化调度模型, 实现生产负载均衡, 并最小化延迟工作量. 另外, 空间众包是面向人力资源的一种调度方式. 针对给定的一组任务集合和工人集合进行任务分配和工人的路径规划. 已有的空间众包调度工作主要聚焦在任务质量控制、工人奖励设置和工人隐私保护等方面[8]. 然而, 以上所述的资源调度技术无法直接应用于 DHRC 任务. 首先, DHRC 任务中的调度需要同时面对设备与人力资源的调度, 并且两者之间的调度存在关联性. 其次, 设备与人力资源之间的协作可能存在顺序依赖, 例如在某个设备执行服务前需要工人先前往某一处获取设备执行所依赖的物品. 最后, 工人的位置随着任务的执行动态变化, 需要预测未来工人的位置. 这些问题对 DHRC 任务中的资源调度提出了挑战.

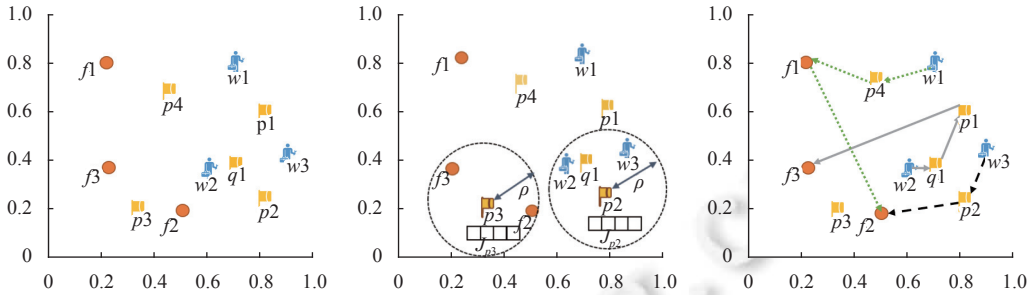
为此, 本文提出一种面向设备与人力资源协作任务的优化调度方法. 该方法分为两个阶段, 首先综合考虑任务距离、设备负载和设备位置周边的工人人数等因素, 提出基于 NSGA-II 的设备资源调度算法, 实现任务对资源的优化选择. 其次根据工人位置和协作依赖等因素, 提出基于 DPSO 的人力资源调度算法, 实现对工人的优化选择以及相应的路径规划. 本文将面对 DHRC 任务的优化调度方法应用于包含 100 个设备、100 个工人、1 000 个任务实例的模拟环境. 实验结果表明本方法能够有效获得调度策略. 另外, 设备资源调度算法在效率上与对比算法相当, 在效用性上优于对比算法. 人力资源调度算法则在效率与效用性上均优于对比算法.

本文第 1 节通过一个动机案例来阐述本文方法的背景和意义. 第 2 节介绍 DHRC 任务及资源调度的相关形式化定义. 第 3 节介绍面向 DHRC 任务的核心调度算法. 第 4 节通过模拟实验对所提方法进行效率和效用性两方面的分析. 第 5 节列举了相关研究工作. 最后是本文的总结和展望.

1 动机案例

假设存在一个由设备、工人和物品资源构成的环境, 如图 1(a) 所示, 包含 3 名工人 $W=\{w1,w2,w3\}$, 4 个设备

资源 $P=\{p1,p2,p3,p4\}$, 以及 1 个提供设备所依赖的物品资源 $Q=\{q1\}$. 设备资源提供服务, 服务被使用后产生相应物品, 设备所在的位置称为设备资源服务点. 物品所在位置称为物品资源服务点, 工人可直接从物品资源服务点处取得物品而无须调用服务. 工人、设备和物品的详情分别如表 1 和表 2 所示, 其中工人均具有取得物品和配送的能力, 每个工人的物品携带量均为 3, 拥有各自的移动速度和位置坐标.



(a) 任务-物理资源-工人空间分布 (b) 物理资源$p2$和$p3$ (c) 经过计算得到的最终结果

图 1 动机案例示意图

表 1 工人详情

工人	能力	携带量	移动速度	初始位置坐标
$w1$	取得物品、配送	3	$v1$	(0.7, 0.8)
$w2$	取得物品、配送	3	$v2$	(0.6, 0.4)
$w3$	取得物品、配送	3	$v3$	(0.9, 0.4)

表 2 物理资源详情

物理资源	类型	关键词	服务时长	位置坐标
$q1$	物品	{u}	—	(0.7, 0.4)
$p1$	设备	{f}	1 min	(0.6, 0.8)
$p2$	设备	{a}	2 min	(0.8, 0.2)
$p3$	设备	{a, b}	{a: 2 min, b: 4 min}	(0.3, 0.2)
$p4$	设备	{c, d}	{c: 3 min, d: 5 min}	(0.5, 0.7)

设备通过不同的关键词表示其提供的服务, 并且每一个服务具有不同的操作时长. 此外, 工人、设备和物品三者的初始位置分布在横坐标与纵坐标区间均为 [0, 1] km 的空间范围内.

在表 2 中, $q1$ 物品资源服务点代表实验室, 学生的学生卡、包裹等物品放置在该位置. 我们将工人在该位置拿取物品的操作定义为关键词 u. $p1$ 代表的终端设备具有开具学生相关证明的服务能力 (f). $p2$ 和 $p3$ 是部署在不同位置的打印机, 均能提供基本的打印文件能力 (a), 另外 $p3$ 打印机还具备彩色打印能力 (b). $p4$ 代表咖啡机, 具有制作普通咖啡 (c) 和拿铁咖啡 (d) 的能力.

案例包含 3 类由用户发起的任务 $Task=\{task1,task2,task3\}$, 任务情况如表 3 所示. 3 个任务中除了包含工人的取得物品和配送任务之外, 还包含线下设备资源需完成的任务. 其中, 任务 $task1$ “递送咖啡”中包含需要拿铁咖啡 (d) 的线下设备资源需求. 任务 $task2$ “会议准备”包含“打印会议文件”和“制作普通咖啡”的 2 个独立的需求. 任务 $task3$ “证明开具”包含 1 个“取学生卡 (u)”和 1 个“学生证明打印 (f)”的需求. 需要注意的是, “取学生卡 (u)”必须要在“学生证明打印 (f)”之前, 即两个服务点位置之间存在前后次序.

表 3 任务详情

任务	名称	最终目的地	关键词	依赖情况
$task1$	递送咖啡	$f1$	{d}	—
$task2$	会议准备	$f2$	{a, c}	—
$task3$	证明开具	$f3$	{u, f}	f → u

在某一时刻, 如图 1(a) 所示, 有 3 个用户分别请求这 3 个任务, 3 个任务都要请求工人将设备生产出物品递送到用户所在处, 最终目的地分别为 $f1$, $f2$ 和 $f3$, 坐标分别为 (0.2,0.8), (0.5,0.2) 与 (0.2,0.4).

从表 3 中不难发现, $task3$ 实例中存在服务点的前后次序约束, 即后一个设备资源的使用依赖工人从前一个物

品资源服务点取得的物品. 传统的针对工人路径规划的调度算法并未考虑该依赖, 可能让工人先前往设备资源 $p1$, 而此时由于并未获取位于 $q1$ 的“学生卡”因此导致设备无法正常执行任务, 进一步带来工人移动成本和时间成本的增加. 另外在实际分配问题中, 存在多个设备资源均提供相同服务能力的情况. 如图 1(b) 所示, 2 台位于 $p2$ 和 $p3$ 的打印机均能提供打印能力, 并且此时两者的负载相同. 传统的调度算法考虑服务点与任务的距离和负载情况两个因素, 考虑到 $p3$ 更接近任务的最终目的地 $f2$, 可能会得出 $p3$ 优于 $p2$ 的调度策略. 然而, 如果选择 $p3$ 的话, 由于在 $p3$ 周围没有工人, 工人相比 $p2$ 会走更远的距离.

基于 DHRC 任务的特殊性, 本文的调度方法加入对服务点顺序约束的考虑, 同时引入设备资源附近顺路工人指标进行设备资源的分配. 将分配后的结果作为工人路径规划算法的输入从而达到优化的目的. 使用本文方法得到的调度结果如图 1(c) 所示.

2 定 义

2.1 基本概念定义

DHRC 任务在人机物融合环境中执行, 涉及设备资源与人力资源等概念. 以下对这些基本概念进行定义.

定义 1. 设备资源 $p=<\phi,dep,l_p,item,\rho,u_p,J_p>$, l_p 为设备资源所处位置, 也称为设备资源服务点. 设备资源提供关键词为 ϕ 服务的资源. 所有设备资源位置集合为 L_p, $l_p\in L_p$. dep 为该资源所依赖的物品输入, 表示该设备只有在具备物品 dep 的情形下才能提供相应的能力以满足服务. $item$ 表示设备资源最终产生的物品. 工人在该点能取得物品 $item$. ρ 表示以 l_p 为圆心的区域半径. u_p 为当前设备资源正在执行的任务, 其剩余执行时间为 $d(u_p)$. J_p 为设备资源 p 的子任务排队队列, 队列的中任务的数量为 $len(J_p)$. 调度平台向队列 J_p 中添加任务, 设备资源 p 按队列顺序依次执行相应任务. 队列中每项子任务 τ 的执行时长为 $d(\tau)$.

定义 2. 物品资源 $q=<\phi,l_q,item>$ 提供关键词为 ϕ 物品, l_q 为物品资源所在位置, 也称为物品资源服务点. 所有物品资源服务点的位置集合为 L_q ($l_q\in L_q$). $item$ 表示该服务点提供的物品, 工人在该点能取得物品 $item$.

定义 3. 人力资源 $w=<l_w,c,J_w,path,v>$ 是具有取物品和配送物品能力的工人. 工人 w 的当前位置为 l_t, 其携带物品的最大数量为 c. 队列中任务的数量为 $len(J_w)$. $path$ 是为工人规划的路径规划子任务执行路径. v 为工人的移动速度.

定义 4. 设备子任务 $\tau_{dv}=<\phi,l_f>$ 是调用设备资源服务并将产出物送至目标位置的任务环节. ϕ 表示对于设备能力的要求 (通过关键词表示). 任务的最终目的地为 l_f, 所有任务最终目的地集合为 L_f ($l_f\in L_f$).

定义 5. 工人子任务 $\tau_{sc}=<l_e,\psi,item,l_f>$ 是由工人完成的物品拿取与递送任务环节. 子任务的执行地点为 l_e, 是产生具体 $item$ 的位置, 并且在设备资源调度后被确定. $\psi=\{\psi-,\psi+\}$ 是工人任务的类型, 其中 $\psi-$ 表示拿取, $\psi+$ 表示递送. 任务最终目的地为 l_f.

定义 6. 必经点 $R=\{r_1,r_2,\ldots,r_m\}$ 是包含 m 个设备资源服务点、物品资源服务点和任务最终目的地的位置集合, 是工人执行取送任务时, 必须访问到的点. 必经点 R 满足 $R\subset L_p\cup L_q\cup L_f$.

定义 7. 路径 $path=[s_w,r_1,r_2,\ldots,r_\theta]$ 为有序序列. 其中, s_w 为工人 w 的路径起始点. r 为必经点, 满足 $r\in R$, 其数量为 θ. 工人 w 经过前 m 个必经点的路径长度 $Distance_w(m)$ 满足:

$$Distance_w(m)=dis(s_k,r_1)+\sum_{i=1}^{m-1}dis(r_i,r_{i+1}),m=1,2,\ldots,\theta \tag{1}$$

其中, $dis(A,B)$ 表示 A 与 B 两个位置的间的欧氏距离. 位置 A 与 B 可以是设备资源服务点、物品资源服务点和任务最终目的地.

定义 8. 设备调度决策 $d_p=<\tau_{dv},p>$ 为设备子任务 τ_{dv} 与设备资源 p 的匹配. 设备调度决策集合 $TP=\{d_p^1,d_p^2,\ldots,d_p^i\}$ 为所有 i 个设备子任务的设备调度决策的集合. 该集合是后续方法框架中第 1 阶段设备资源调度算法的输出和第 2 阶段人力资源调度算法的输入.

定义 9. 工人调度决策 $d_w=<\tau_{sc},r,w>$ 为工人子任务、必经点与工人的匹配, 包含了所分配工人的路径规划.

集合 $SR=\{d_w^1,d_w^2,\ldots,d_w^i\}$ 为所有 i 个工人调度决策的集合, 所有工人的路径集合形成工人路径规划表 WL, WL 是后续方法框架中第 2 阶段人力资源调度算法的输出.

2.2 设备调度决策相关定义

在 DHRC 任务的调度过程中, 设备调度是一个重要环节. 该环节涉及以下的决策变量、目标函数和约束条件方面的定义.

(1) 决策变量

针对 t 时刻新产生的 n 个设备子任务的集合, 定义决策变量.

$$x_{i,j}=\begin{cases}1, \text{第}i\text{个设备子任务分配到第}j\text{个设备资源上}\\ 0, \text{否则}\end{cases} \tag{2}$$

(2) 目标函数

$$\min Z=\min\{G(x_{i,j}),H(x_{i,j})\} \tag{3}$$

目标函数 Z 是 $G(x_{i,j})$ 和 $H(x_{i,j})$ 两部分成本的组合优化, 目的是取得最小值. $G(x_{i,j})$ 表示分配后的时间成本, 由配送时间成本 $f(x_{i,j})$ 和等待时间成本 $g(x_{i,j})$ 组成, 即 $G(x_{i,j})=f(x_{i,j})+g(x_{i,j})$.

配送时间成本 $f(x_{i,j})$ 就是将物品从设备资源配送到应用最终目的地的时间. 如果要让配送时间成本越少, 那么所选的设备资源位置就必须越靠近应用的最终目的地. 综上, 构造配送时间成本 $f(x_{i,j})$.

$$f(x_{i,j})=\frac{1}{v}\times\sum_{i=1}^{n}\sum_{j=1}^{m}x_{i,j}\times dis(p_j,\tau_{dv}.l_f) \tag{4}$$

其中, $f(x_{i,j})$ 为工人以平均速度 v 从设备资源移动到最终目的地所需花费的时间.

在实际场景中, 应用最终目的地往往集中在某个楼宇内, 如: 实验楼、图书馆. 如果仅考虑设备资源与应用最终目的地之间的距离因素, 存在很大可能会将任务分配到距离这些楼宇较近的固定几个设备资源, 而忽略其他空闲但距离稍远的设备资源的情况. 造成由于不均衡的分配导致某些设备资源的任务过多从而使得等待时间过长. 为了避免产生上述不均衡分配的情形, 本文考虑了使用设备资源的任务队列长度作为等待时间成本来进行优化. 在第 j 个设备资源上, 其队列 J_{p_j} 中的前 n 个子任务理论完成的时长为:

$$L(n,p_j)=d(u_{p_j})+\sum_{k=1}^{n}d(\tau_{dv}^k) \tag{5}$$

等待时间成本为:

$$g(x_{i,j})=L(len(J_{p_j}),p_j)+x_{i,j}\times d(\tau_i) \tag{6}$$

$H(x_{i,j})$ 为设备资源附近顺路工人指标. 在面向设备与人力资源协作任务中, 设备资源和人力资源之间的协作尤为重要. 在进行设备子任务的资源分配时, 如果能同时考虑设备资源周围的顺路工人因素, 那么就会降低设备资源周围缺少工人的情况发生概率.

定义 10. 以设备资源 $p.l_p$ 所在位置为圆心, 在半径 $p._\rho$ 范围内的区域称为服务区域, 记作 $Area_p$. 如果工人恰好在 $Area_p$ 区域内, 即 $dis(p.l_p-w.l_w)\leqslant radius$, 则称该工人为 $Area_p$ 区域的顺路工人. $Area_p$ 区域内顺路工人的数量为 C_p.

对所有设备资源, 顺路工人的满足:

$$H(x_{i,j})=\sum_{j=1}^{m}\frac{\sum_{i=1}^{n}x_{i,j}}{C_j} \tag{7}$$

(3) 约束条件

$$\tau_{dv}.\phi\subseteq p.\Phi \tag{8}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}x_{i,j}=n, i=1,2,\ldots,n, j=1,2,\ldots,m \tag{9}$$

$$x_{i,j}\geqslant 0, i=1,2,\ldots,n, j=1,2,\ldots,m \tag{10}$$

其中, 公式 (8) 是设备资源能力约束, 待分配的设备资源必须能够满足设备子任务所需的能力要求, 即需满足特定的能力属性集合 (记为 $p.\Phi$). 具体而言, 当决策变量 $x_{i,j}=1$ 时, 表示第 i 个设备子任务 $\tau_{dv,i}$ 分配给第 j 个设备资源 p_j, 此时必须保证设备子任务 $\tau_{dv,i}$ 所需能力 $\tau_{dv,i}.\phi$ 的集合是设备资源 p_j 提供的服务集合 $p_j.\phi$ 的子集. 公式 (9) 保证所有 n 个设备子任务都被分配到合适的设备资源. 公式 (10) 是对于决策变量取值范围的约束.

2.3 工人调度决策相关定义

人力资源的调度也是 DHRC 任务调度过程中的重要环节. 人力资源调度的目标是选择工人并且进行路径规划, 是典型的由分布在环境中的多工人、多任务组成的动态配送问题 (dynamic pickup and delivery problem, DPDP). 该问题具有以下定义.

定义 11. 时间 $t_w^{\text{wait}}(m)$ 为工人 w 在必经点 r_m 处的等待时间.

工人 w 在物品资源服务点和最终配送目的地无需等待, 此时 $t_w^{\text{wait}}(m)=0$. 工人 w 在设备资源需要在物品产出之后才能取得相应物品, 此时等待时间有两种情况, 第 1 种是工人到达时间早于任务执行完成时间, 这种情况下工人需要等待任务执行完成. 另一种工人到达时间晚于任务执行完成时间, 此时工人无需等待. 综上, 等待时间满足:

$$t_w^{\text{wait}}(m)=\begin{cases}\max\{0,t_w^{\text{end}}(m)-t_w^{\text{arr}}(m)\}, & \text{必经点为设备资源}\\ 0, & \text{否则}\end{cases} \tag{11}$$

在 r_m 为设备资源的条件下, $t_w^{\text{end}}(m)$ 为工人 w 离开设备资源 r_m 的时刻. 忽略工人从必经点拿取物品的时间, 工人离开设备资源 r_m 的时刻就是对应第 n 个设备子任务 τ_{dv} 在必经点的执行结束时刻, 满足:

$$t_w^{\text{end}}(m)=L(n,r_m)=d(u_{r_m})+\sum_{k=1}^{n}d(\tau_{dv}^k) \tag{12}$$

$t_w^{\text{arr}}(m)$ 为工人 w 到达必经点 r_m 的时刻. 工人 w 到达必经点 r_m 的时刻满足:

$$t_w^{\text{arr}}(m)=t_w^{\text{end}}(m-1)+dis(r_{m-1},r_m)/v \tag{13}$$

下面给出面向设备与人力资源协作任务中, 关于 DPDP 问题的前提假设、目标函数和约束条件. 在 t 时刻, 针对所有 $K(t)$ 个在线的工人.

(1) 前提假设

① 设备资源子任务之间的切换时间忽略不计.

② 工人从必经点拿取物品和递交物品的时间忽略不计.

(2) 目标函数

$$\min Cost=\sum_{k\in K(t)}Distance_w(m)/v+\sum_{k\in K(t)}\sum_{\forall r_m\in R(k)}t_w^{\text{wait}}(m) \tag{14}$$

目标函数是为了最小化执行时间成本 $Cost$. $Cost$ 由工人在必经点间移动时间和工人在必经点的等待时间两部分组成.

(3) 约束条件

对于人力资源调度路径规划, 所需满足如下约束.

① 物品约束, 同一任务下的路径规划子任务, 只能由 1 个工人执行, 但工人可接收并依次执行多个路径规划子任务.

② 顺序约束, 同一个物品的取动作必须在送动作之前发生.

③ 工人容量约束, 物品装载量不能超过工人可携带的最大容量.

3 面向 DHRC 任务的优化调度算法

3.1 调度框架

面向 DHRC 任务的调度框架如图 2 所示. 调度框架主要由工人位置估计、设备资源调度和人力资源调度这 3 大模块组成. 其中, 工人位置估计模块负责估计应用中的设备子任务在未来执行时工人位置在空间中的分布情

况, 其输入为当前实时的工人位置信息和工人路径规划表. 设备资源调度模块根据物理设备的空间分布, 当前的任务数量和工人空间分布将任务分配到合适的设备资源上. 人力资源调度模块根据工人的位置和技能、物品与设备信息进行工人的选择以及任务的分派, 并且规划出所选择工人的行进路径.

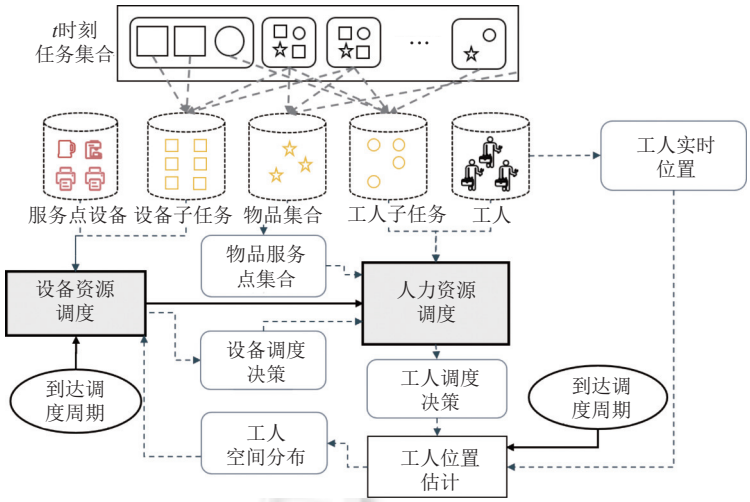


图 2 面向 DHRC 任务的调度框架

在一个调度周期内, 调度框架将获取该周期内用户提交的任务中所有的子任务信息. 对子任务进行分析, 进而得到设备子任务和工人子任务这两种类型的子任务以及各子任务之间的先后顺序关系. 设备子任务作为设备资源调度算法的输入, 工人子任务作为人力资源调度算法的输入. 随后, 调度框架执行相应调度动作, 最终得到设备子任务的设备调度决策和工人子任务的工人调度决策, 其包含工人选择及工人的路径规划表.

3.2 算法设计

3.2.1 设备资源调度

设备资源的调度是典型的组合优化问题. 常用的求解算法有枚举、分支界定等精确方法, 贪心等近似方法, 神经网络, 以及遗传算法、蚁群算法、进化算法等启发式算法. 对于设备资源的调度来说, 需要综合考虑设备负载、与任务的距离, 工人的空间分布等情况.

NSGA-II 算法是一种启发式算法, 它是 Deb 等人在原有 NSGA 算法基础上进行的改进[9], 提升了算法的多目标组合优化的收敛性和多样性. NSGA-II 算法将传统的遗传算法 (GA) 与 Pareto 最优的概念进行结合. 在传统遗传算法中的排序部分, 使用帕累托最优 (Pareto optimality) 概念中的非支配排序作为新的排序方式. 在个体选择部分, 使用基于参考点的方法作为新的个体选择方法. 在种群更新部分, 使用精英策略生成下一代种群, 提高种群的多样性.

NSGA-II 算法的主要特点是具有较好的全局搜索能力, 适用于解决具有双重目标的组合优化问题. 因此本文使用 NSGA-II 算法进行求解.

(1) 算法流程

算法 1 是设备资源调度算法的伪代码. 算法首先根据任务所需能力和设备所提供能力, 先进行过滤, 得到满足条件的设备列表. 接着进行种群的初始化和选择操作. 然后开启算法的迭代. 在每一轮迭代中, 进行 Step 1 将染色体进行交叉操作; Step 2 将染色体进行变异操作; Step 3 将原始种群与经过交叉、变异后的种群进行合并; Step 4 在合并后的种群中使用精英策略选择优秀的种群保留到下一次迭代. 在满足退出条件时, 选出最优基因并进行解码, 最终得到设备调度决策 *TP* 作为算法的返回值. 算法退出条件为达到迭代次数上限或在一定次数内目标函数值的几乎不变.

(2) 遗传编码设计

NSGA-II 算法中的染色体编码由决策变量 x 构成. 考虑到决策变量组成的 0-1 矩阵是稀疏矩阵, 直接进行编码会导致染色体过长, 计算量偏大. 本文采用正整数的一维有序数组方式进行染色体编码. 染色体长度等于设备子任务的总数 m. 基因由按任务编号顺序排列的设备资源服务点分配编号组成, 表示一种可能的分配方案. 为满足每一批任务必须分配一个设备资源服务点的约束条件, 染色体长度为当前待任务的数量 n. 基因的取值范围是 [1, m]. 当设备资源服务点数量 m=4, 任务数 n=5 时, 染色体编码如图 3.

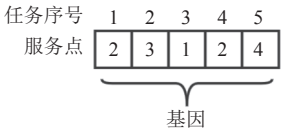


图 3 设备资源服务点染色体编码

图 3 中的染色体使用 1×5 大小的数组进行编码, 数组从左到右分别表示第 2 个设备资源服务点分配给第 1 个设备子任务, 第 3 个设备资源服务点分配给第 2 个设备子任务, 第 1 个设备资源服务点分配给第 3 个设备子任务, 第 2 个设备资源服务点分配给第 4 个设备子任务, 第 4 个设备资源服务点分配给第 5 个设备子任务.

(3) 适应度函数

本文使用第 2.2 节中的目标函数 (即公式 (3)) 进行 NSGA-II 算法中的适应度函数设计. 值得注意的是, 由于目标是求最小化问题, 因此需要对目标函数进行变换, 使得目标函数的数值越小适应度越大, 目标函数的数值越大适应度越小.

此外, 设备调度算法还需要满足其约束条件, 即公式 (8)–公式 (10). 此时设定惩罚值, 当约束条件不被满足时, 适应度函数的值理论上应取负无穷大. 综合考虑上述情况, 设计适应度函数:

$$fitness(x)=\begin{cases}-z, & \text{满足约束条件}\\ -INF, & \text{否则}\end{cases} \tag{15}$$

(4) 遗传算子设计

在 NSGA-II 算法中, 首先进行初始化操作, 生成初代种群 (population). 然后对该种群进行迭代, 直到满足算法退出条件得到最终结果. 算法退出条件为达到迭代次数上限或在一定次数内目标函数值的几乎不变. 每次迭代需要依次进行选择 (*SelectPopulation*)、交叉 (*Crossover*)、变异 (*Mutation*) 和合并 (*Merging*) 操作. 这些操作均由对应的算子实现, 下面是算子的详细设计.

① 选择算子

• Step 1. 判断种群是否为初始种群, 若是则生成数量为 N 的种群, 并对种群进行排序. 否则执行 Step2.

• Step 2. 对上一阶段的合并后的种群按适应度进行排序.

• Step 3. 从父代与子代融合的群体中选择 N 个个体, 作为新的迭代种群.

② 交叉算子

如图 4 所示, 交叉操作一共分为 3 个步骤.

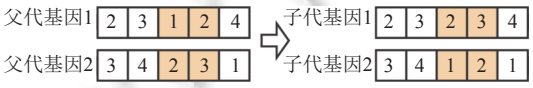


图 4 设备资源服务点染色体交叉

• Step 1. 从迭代种群中随机选取 K_c 个个体, K_c 满足小于等于种群数 N 且 K_c 为偶数. K_c 个个体两两组合形成 $K_c/2$ 个基因对.

• Step 2. 针对每个基因对, 随机生成 2 个范围为 [1,n] 的整数, 作为交叉点.

• Step 3. 循环遍历所有基因对, 交换交叉点的数值, 完成交叉操作, 形成新的基因对. 循环结束后, 最终产生 K_c

个新的交叉后基因种群.

③ 变异算子

如图 5 所示, 变异操作一共分为 3 个步骤.

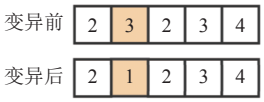


图 5 设备资源服务点染色体变异

• Step 1. 从迭代种群中随机选取 K_m 个个体, K_m 满足小于等于种群数.

• Step 2. 针对每个基因, 随机生成 1 个范围为 $[1,n]$ 的整数, 作为变异点.

• Step 3. 循环遍历所有基因, 修改每个基因中的变异点为新的值 Z. Z 为随机生成的范围为 $[1,m]$ 的整数. 循环结束后, 最终产生 K_m 个新的变异后基因种群.

④ 合并算子

合并算子是将上述交叉后基因种群, 变异后基因种群和迭代种群进行合并, 形成新的合并后种群. 合并后种群规模大于迭代种群.

算法 1. 基于 NSGA-II 的设备资源调度算法.

输入: T_{dv}, P, W;

输出: TP.

$TP \leftarrow \{\}$

$WorkerMap \leftarrow \{\}$

$P_\phi \leftarrow \{p | \forall p \in P, \tau.\phi \cap p.\Phi \neq \varnothing\}$

FORALL $p \in P_\phi$ **DO**

$WorkerMap.append(positionEstimate(W, getTime(len(J_p))))$

ENDFOR

$popInit \leftarrow InitPopulation(encodeGene(T_{dv}, P_\phi, WorkerMap))$

$pop \leftarrow SelectPopulation(popInit)$

WHILE *iter < Maximum Number Of Iterations* **Or** *fitness no longer updates* **DO**

$popCrossed \leftarrow Crossover(pop)$

$popMutated \leftarrow Mutation(pop)$

$popMerged \leftarrow Merging(pop, popCrossed, popMutated)$

$pop \leftarrow SelectPopulation(popMerged)$

ENDWHILE

$popSorted \leftarrow sortByFitness(pop)$

$geneBest \leftarrow getBestGene(popSorted)$

$TP \leftarrow decodeGene(geneBest)$

RETURN TP

设备资源调度算法中, 首先读取设备子任务 T_{dv}, 设备资源集合 P 和总工人 W 作为输入. 接着, 根据设备子任务 T_{dv} 和设备资源集合 P 的关键词, 匹配得到满足条件的设备资源 P_ϕ, 并根据每个设备资源的队列情况, 分别估计未来队列任务完成时工人的位置, 构建 *WorkerMap*. 接着调用 *InitPopulation* 生成初始种群, 并调用 *SelectPopulation* 进行排序选优. 然后进行循环迭代, 直到达到迭代次数或适应度不再更新的退出条件. 在每个循环内, 依次调用交

叉算子 (*Crossover*), 变异算子 (*Mutation*) 对本轮迭代中原始种群中的染色体迭代进行更新. 同时调用 *Merging* 函数合并本轮迭代中原始种群 pop、交叉后的种群 *popCrossed* 和变异后的种群 *popMutated*. 最后调用 *SelectPopulation* 进行排序选优, 更新原始种群 pop. 结束迭代后, 从种群 pop 中找到最优的染色体进行解码, 最终输出设备调度决策 TP.

3.2.2 工人位置估计

在实际场景中, 资源调度是一个动态的过程. 随着任务的进行, 场景中的工人空间位置不断发生变化. 而在给设备子任务分配合适的服务点时, 又必须考虑任务执行完成时服务点周围工人的空间分布情况. 因此, 需要对未来时刻的工人的位置进行估计. 文献 [10] 中提出了基于对于未来工人和任务的预测方法用于工人子任务的分配. 而本文考虑工人估计是为了更好的辅助设备子任务的分配, 对预测精度要求不高. 因此, 本文提出一种简单、高效的工人位置估计算法, 即算法 2.

算法 2. 工人位置估计算法.

输入: W, t_f;

输出: L_w.

$L_w \leftarrow \{\}$

FORALL $w \in W$ **DO**

$\delta (\leftarrow t_f - t)$

$s_\delta \leftarrow \delta \times v$

$path_{org} \leftarrow w.path$

$i \leftarrow$ 沿路径 $path_{org}$ 找到满足 $Distance_w(i) > s_\delta$ 且 $Distance_w(i-1) < s_\delta$ 的 i

$s_{offset} \leftarrow s_\delta - Distance_w(i-1)$

$direction \leftarrow$ 根据 r_{i-1}, r_i 两个位置, 获取方向

$l_w \leftarrow r_{i-1} + direction \times s_{offset}$

$L_w.append(l_w)$

ENDFOR

RETURN L_w

在当前 t 时刻, 算法的输入为当前工人集合 W 和未来 t_f 时刻 ($t_f > t$). 输出为工人在地图上的位置分布, 记作 L_w.

工人位置估计算法根据工人的移动路径 $path$ 进行估计. 具体而言, 首先计算出工人沿 $path$ 行走的区间, 根据未来时间 t_f 确定工人在该区间内行进的距离和方向, 从而估计出工人的位置, 使得设备资源调度能更加精准.

3.2.3 人力资源调度

对于人力资源的调度实际上是一个空间众包调度问题. 人力资源调度是 NP-Hard 问题. 求解此类问题有禁忌搜索、粒子群优化、遗传算法等启发式算法. 禁忌搜索算法具有局部开发能力强, 收敛速度快的特点, 但是其容易陷入局部最优. 传统的遗传算法虽然较强的全局搜索能力, 但其求解效率相对较低. 传统粒子群优化虽然搜索速度快、算法简单, 但对于此类离散型优化问题效果不佳, 也容易过早收敛到局部最优. 空间工人子任务分配和路径规划来说, 尽管需要综合考虑工人移动时间和工人等待时间等情况, 但不涉及多目标优化问题, 因此本文采用离散粒子群优化算法进行求解, 同时针对其编码方式和算子进行改进.

(1) 算法流程

算法 3 是人力资源调度算法的伪代码. 首先根据工人子任务 T_{sc}、必经点 R 和工人 W 初始化粒子群. 接着开启算法的迭代. 在每一轮迭代中, 进行 Step 1 粒子位置向量更新; Step 2 粒子位置向量变异; Step 3 适应度计算; Step 4 局部和全局最优值更新. 在达到迭代次数后, 返回最优粒子所包含的工人路径.

算法 3. 基于 DPSO 的人力资源调度算法.

输入: T_{sc}, R, W;

输出: WL.

$WL \leftarrow \{\}$

$x \leftarrow initPopulation(T_{sc}, R, W)$

$iter \leftarrow 0$

WHILE $iter < Maximum\ Number\ Of\ Iterations$ **Or**

$fitness\ no\ longer\ updates$ **DO**

$x \leftarrow updatePosition(x, pBest, gBest)$

$x \leftarrow Mutation(x)$

$fitness \leftarrow calculateFitness(x)$

$pBest, gBest \leftarrow updateBest(fitness, pBest, gBest)$

$iter \leftarrow iter + 1$

ENDWHILE

$particle \leftarrow getParticle(x, gBest)$

$WL \leftarrow decodeParticle(particle)$

RETURN WL

(2) 粒子编码

本文算法将粒子编码为一个 D 维的向量, 向量的每一维由一个结构体描述. 结构体由工人编号, 工人子任务编号, 必经点编号, 必经点类型和工人子任务类型这 5 种要素组成. 其中, 工人子任务编号是当前调度周期内所有的工人子任务的编号, 取值范围是 [1, N] 区间的整数, N 为工人子任务个数. 依照之前的假设, 每个工人子任务只能由 1 个工人执行, 因此每条染色体中工人子任务不重复, 即染色体的长度为 N. 工人子任务类型的取值有取物品 $\psi-$ 和送物品 $\psi+$ 两种. 必经点编号取值范围是 [1, M] 区间的整数, M 为必经点个数. 对于每个位置向量, 需要包含所有 M 个不同的必经点, 同时允许存在重复必经点的情况. 必经点类型的取值有设备资源服务点 P, 物品资源服务点 Q 和最终目的地 F 这 3 种. 其数量分别为 M_p, M_q 和 M_f. 服务点数量满足 $M_s = M_p + M_q$, 必经点数量满足 $M = M_s + M_f$. 粒子群初始化时, 生成数量一定、位置随机的粒子. 随着算法的迭代, 粒子的位置向量也会随之更新. 为了便于后续算法的描述, 在不产生歧义的前提下, 后续使用工人子任务编号指代对应的粒子的位置向量.

图 6 中, 展示了粒子的位置向量编码. 其中, 工人的数量为 3, 工人子任务个数为 11, 编号为 1−11. 必经点个数为 10. 其中, 服务点的编码集合为{1,2,3,4,5}, 总数量 M_s 为 5. 在所有服务点中, 物品资源服务点编码为{1,2}, 数量 M_q 为 2, 类型为 Q. 设备资源服务点编码为{3,4,5}, 数量 M_p 为 3, 类型为 P. 最终目的地的编码集合为{6,7,8,9,10}, 数量 M_f 为 5, 类型为 F.

工人编号	1				2					3		
工人子任务编号	10	4	5	11	8	7	3	2	6	9	1	工人子任务
必经点编号	5	6	8	7	1	2	9	4	10	3	7	必经点匹配
必经点类型	P	F	F	F	Q	Q	F	P	F	P	F	
工人子任务类型	ψ^-	ψ^+	ψ^+	ψ^+	ψ^-	ψ^-	ψ^+	ψ^-	ψ^+	ψ^-	ψ^+	

图 6 粒子的位置向量编码

工人 1 执行任务顺序为 $10 \rightarrow 4 \rightarrow 5 \rightarrow 11$, 路径为 $5 \rightarrow 6 \rightarrow 8 \rightarrow 7$. 工人 2 执行任务顺序为 $8 \rightarrow 7 \rightarrow 3 \rightarrow 2 \rightarrow 6$, 路径为 $1 \rightarrow 2 \rightarrow 9 \rightarrow 4 \rightarrow 10$. 工人 3 执行任务顺序为 $9 \rightarrow 1$, 路径为 $3 \rightarrow 7$.

(3) 适应度函数

在人力资源调度中, 本文使用第 2.3 节中的目标函数 (即公式 (14)) 来进行本文算法中的适应度函数设计, 适应度越大越好. 而目标函数是描述时间成本的函数, 其数值越小越好. 因此, 在实际计算时, 取目标函数值的相反数, 使得目标函数的数值越小, 适应度越高. 对于不满足约束条件的函数, 其适应度为负无穷大. 综合考虑上述情况, 设计适应度函数:

$$fitness(x)=\begin{cases}-Cost, & \text{满足约束条件}\\ -INF, & \text{否则}\end{cases} \tag{16}$$

(4) 粒子位置更新

在粒子群优化算法中, 粒子的位置需要根据历史最优 $pBest$ 和粒子群最优 $gBest$ 进行不断更新迭代. 在本文的算法中, 粒子的更新策略如公式 (17) 所示. 其中, t 表示当前迭代次数, w 表示惯性权重, c_1, c_2 表示学习因子, 并且满足 $c_1+c_2=1$. r_1 表示取值范围 [0, 1] 之间的随机数.

$$x^{t+1}=\begin{cases}Crossover(x^t,pBest^t), & r_1<\omega \text{ and } r_1<c_1\\ Crossover(x^t,gBest^t), & r_1<\omega \text{ and } r_1\leqslant c_2\\ Perturbation(x^t), & r_1\geqslant\omega\end{cases} \tag{17}$$

粒子位置的更新同样由对应的算子实现, 下面是算子的详细设计.

① 交叉算子

图 7 描述了交叉算子 (*Crossover*) 更新粒子的位置向量的过程. 针对迭代过程中的每一个粒子其具体的交叉步骤如下.

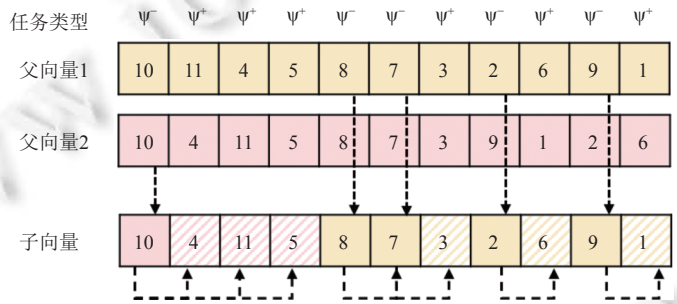


图 7 粒子群优化交叉算子

• Step 1. 判断 r_1 与 c_1 的大小, 当 $r_1<c_1$ 时, 进入 Step 2; 否则, 进入 Step 4.

• Step 2. 创建两个父代向量 *parent*1 和 *parent*2 分别记录当前粒子和历史最优 $pBest^t$ 的粒子位置. 并创建子代向量 *child*.

• Step 3. 按顺序遍历 *child*, 当 *child* 的维度对应的工人子任务类型为“取得物品”时, 随机从向量 *parent*1 和 *parent*2 的对应维度中选取工人编号填入 *child* 中. 并将该工人编号写入 *child* 中对应工人子任务类型为“配送物品”的维度中.

• Step 4. 类似地, 创建两个父代向量 *parent*1 和 *parent*2 分别记录当前粒子和粒子群最优 $gBest^t$ 的粒子位置. 并创建子代向量 *child*.

• Step 5. 按顺序遍历 *child*, 当 *child* 的维度对应的工人子任务类型为“取得物品”时, 随机从向量 *parent*1 和 *parent*2 的对应维度中选取工人编号填入 *child* 中. 并将该工人编号写入 *child* 中对应工人子任务类型为“配送物品”的维度中.

② 扰动算子

为了提高粒子群的多样性, 避免过早陷入局部最优. 算法对粒子的位置向量进行一定程度的扰动 (*Perturbation*). 针对迭代过程中的每一个粒子其具体扰动步骤如下.

• Step 1. 随机从工人子任务类型为“取得物品”的粒子位置向量中选择 m 个维度作为扰动点, m 满足不超多粒子的位置向量长度 D.

• Step 2. 针对每个扰动点, 将该扰动点工人与任务数最少的工人进行比较, 如果该扰动点工人任务数超过最少任务数则将该点工人编码扰动为最少任务数工人的编码. 如果有多个最少任务的工人, 则随机选取一个工人进行替换.

• Step 3. 遍历粒子位置向量, 更新对应工人子任务类型为“配送物品”的维度的工人编码.

③ 变异算子

影响人力调度中的时间成本除了每个工人子任务分配工人之外, 还有每个工人的路径. 因此本文设计变异算子 (*Mutation*) 来针对工人的路径进行优化, 如图 8 所示. 具体变异步骤如下.

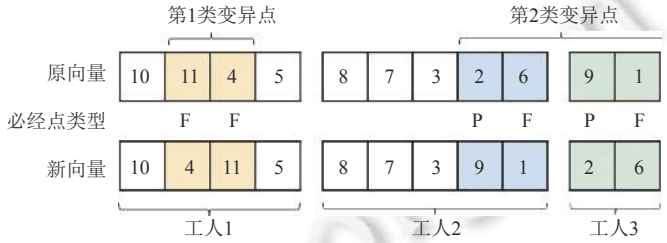


图 8 粒子群优化变异算子

• Step 1. 从迭代粒子群中随机选取 K_m 个个体, K_m 满足小于等于粒子群数.

• Step 2. 针对每个位置向量, 随机选取长度为 2 的位置片段作为变异点. 如果两个变异点的必经点类型均为最终目的地, 即第 1 类变异点, 则直接交换两个变异点. 如果两个变异点中前一个为服务点, 后一个为最终目的地, 即第 2 类变异点, 则从再从染色体中找到第 2 类变异点, 将两个位置片段进行交换. 如果两个变异点中, 前一个为最终目的地, 后一个为服务点, 或者两个点均为服务点, 则重新随机选取变异点.

• Step 3. 循环遍历所有位置向量, 最终产生 K_m 个新的变异后粒子群集合.

人力资源调度算法中, 首先读取工人子任务 T_{sc}, 必经点 R 和总工人 W. 通过 *initPopulation* 函数随机生成一定数量的初始种群 x. 每个种群由粒子组成, 其编码方式如图 6 所示. 接着进行循环迭代, 直到达到迭代次数或适应度不再更新的退出条件. 在每个循环内, 依次调用 *updatePosition* 进行粒子位置向量更新, 调用 *Mutate* 进行粒子位置向量变异, 调用 *calculateFitness* 和 *updateBest* 更新粒子适应度的局部和全局最优值. 最后, 从种群 x 中选取全局最优对应的粒子. 最终通过 *decodeParticle* 将粒子解码为工人路径规划表 *WL* 并输出.

4 实验与分析

为评估所提出调度方法的性能, 本文进行了一系列模拟实验, 从以下 3 个维度的指标进行实验验证.

• RQ1 (算法的有效性): 算法是否能够满足面向 DHRC 任务的调度需求?

• RQ2 (算法的效率): 算法本身的计算是否高效?

• RQ3 (其他参数对算法的影响): 其他参数设置对实验结果是否有影响?

4.1 实验设计

4.1.1 实验环境

本文人机物融合应用的调度框架部分使用 Python 进行编写, 使用 MySQL 存储与实验相关的静态和运行时的数据. 调度框架部署在 Ubuntu 20.04 的服务器上. 服务器的 CPU 为 Intel® Xeon® Gold 6226R CPU@2.90 GHz, 内存大小为 64 GB.

4.1.2 实验流程

模拟实验由模拟实验引擎负责具体实验的实施, 其完整的流程如图 9 所示. 首先, 模拟实验引擎读取实验配置文件, 设置调度周期和模拟实验总批次, 初始化服务点, 工人数量和位置. 初始化迭代次数、种群规模等算法参数.

其次, 进行周期循环直到达到实验总批次. 在每个调度周期内, 从数据库中按顺序读取一定数量的应用实例, 作为调度任务输入到调度框架中, 开启两阶段调度算法. 在每个实验批次内, 更新设备资源服务点的设备子任务队列以及工人的位置和工人子任务队列.

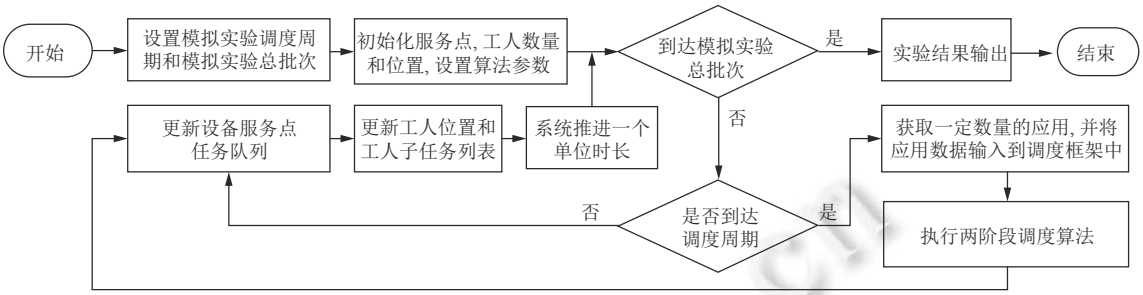


图 9 模拟实验的具体流程

4.1.3 实验数据

本实验旨在通过模拟场景进行研究. 实验区域设置在横坐标和纵坐标范围为 0–1 km 的空间内. 首先, 为了模拟现实世界中设备资源服务点的数量和分布情况. 本文选择了上海市多所高校作为中心, 并以半径 1 km 内的店铺 (视作为本文方法中的设备资源服务点) 进行过滤和统计, 得到真实世界店铺的数量. 同时, 采用正态分布的模型计算了 1 km 范围内主要店铺坐标的分布参数. 其次, 为了模拟现实世界中工人的情况, 本文从蜂鸟众包和美团骑手等众包应用在各个高校周边 1 km 内的平均数量调研, 作为工人数量的参考依据. 根据真实世界中店铺和工人的统计数据, 本文利用 gMission 平台提供的数据生成工具创建了一个包含 100 个工人、1 000 个应用实例、100 个设备资源服务点和 50 个物品服务点的测试数据集. 这些数据集被存储在数据库中, 并且每个周期从数据库中获取一定数量的应用实例, 将其输入到调度框架中进行处理. 本文假设在初始条件下工人均匀分布在实验区域内, 随着实验的进行, 工人的位置会根据分配的任务情况来进行移动. 假设工人的平均移动速度为 18 km/h. 此外, 本文设定了 4 种类型的服务, 分别为“制作咖啡”“制作食品”“打印文件”“烧水”, 每个设备资源服务点提供其中一种服务. 同时, 每个设备资源在同一时间只能执行一项设备任务, 后续任务将进入该设备的子任务排队队列. 此外, 物品服务点包含了 6 种不同类型的物品.

4.1.4 对比算法

由于调度框架涉及两个阶段的调度过程. 因此需要分别评估每个过程的算法. 两个阶段的算法在两个维度进行对比, 分别是算法的执行时长 (Time) 和算法计算得出的调度成本 (*Cost*).

为了避免名称混淆, 本文将阶段 1 的本文所采用的设备资源调度算法称为 DRSA, 将阶段 2 所采用的人力资源调度算法称为 HRSA. 将本文在设备资源调度阶段算法 (DRSA) 与离散粒子群优化算法 (DPSO) 进行对比. 将本文在人力资源调度阶段算法 (HRSA) 与使用锦标赛机制改进的遗传算法 (MGA) 进行对比.

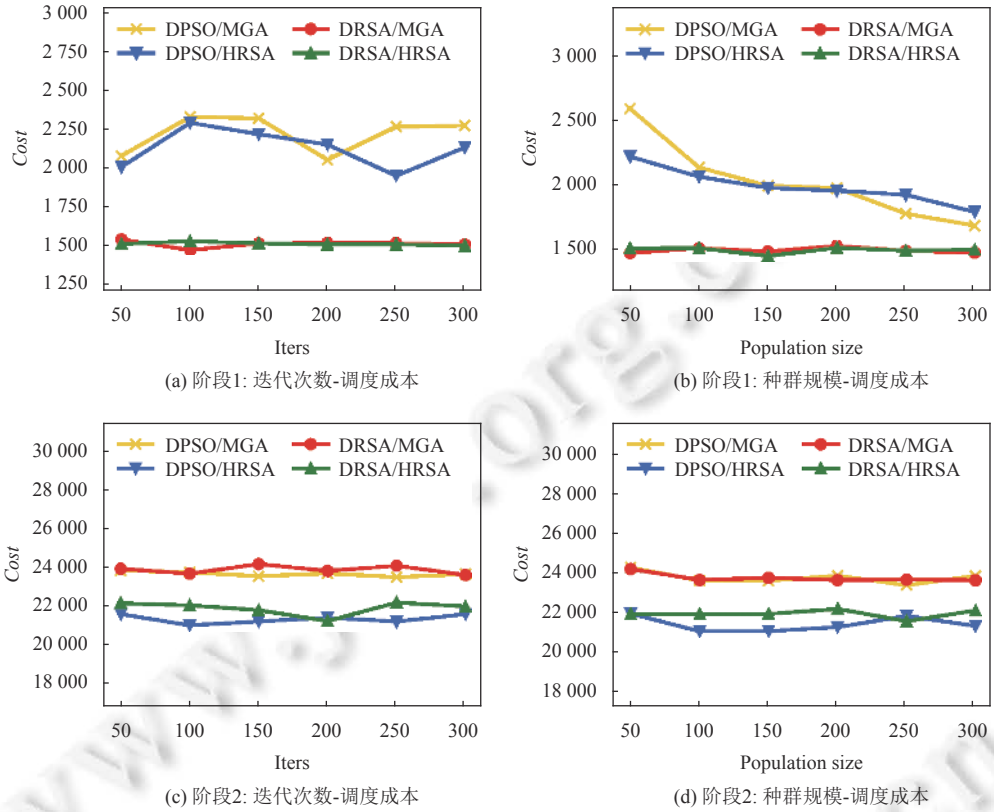
4.2 实验结果与分析

● RQ1: 算法的有效性

算法的效率实验目的是对两阶段算法的最终调度成本进行评估. 具体而言, 首先保持的阶段 2 的种群规模和迭代次数不变, 分别改变阶段 1 的设备资源调度算法的迭代次数和种群规模并统计两阶段算法的调度成本之和. 此时阶段 2 的两种算法迭代次数均设置为 100 次, 种群规模均设置为 100. 其次, 保持阶段 1 的种群规模和迭代次数不变, 分别改变阶段 2 的人力资源调度算法的迭代次数和种群规模并统计两阶段算法的调度成本之和. 此时阶段 1 中两种算法迭代次数均设置为 100 次, 种群规模设置为 100. 实验运行 3 个周期, 保持每个周期输入调度框架的应用数量为 100 个. 设备资源服务点数量设置为 100, 工人数量设置为 20.

图 10(a) 展示了迭代次数对于阶段 1 设备资源调度算法调度成本 *Cost* 的影响. 横坐标为阶段 1 算法的迭代次数. 图 10(b) 展示了初始种群规模对于阶段 1 设备资源调度算法调度成本 *Cost* 的影响. 横坐标为阶段 1 算法的初

始种群规模. 图 10(c) 展示了迭代次数对于阶段 2 人力资源调度算法调度成本 *Cost* 的影响. 横坐标为阶段 2 算法的迭代次数. 图 10(d) 展示了初始种群规模对于阶段 2 人力资源调度算法调度成本 *Cost* 的影响. 横坐标为阶段 2 算法的初始种群规模.



(a) 阶段1: 迭代次数-调度成本 (b) 阶段1: 种群规模-调度成本

(c) 阶段2: 迭代次数-调度成本 (d) 阶段2: 种群规模-调度成本

图 10 RQ1: 算法有效性实验结果

从图 10(a) 中可以看出, 随着迭代次数的增大, 阶段 1 设备资源调度算法的对比算法 (蓝色和黄色折线) 的调度成本 *Cost* 值先增大后减小. 当阶段 2 采用本文算法 HRSA (蓝色折线) 时, *Cost* 值下降更明显. 而图 10 中红色和绿色折线所代表的阶段 1 的本文算法 DRSA 的调度成本 *Cost* 值缓步下降, 两条折线的 *Cost* 平均值从 1 550 降低到 1 508. 从数值上看, 无论阶段 2 使用哪种算法, 阶段 1 中本文算法 DRSA 较对比算法效果更好. 这是因为对比算法容易陷入局部最优, 多次迭代实验导致其没有收敛到接近最优解. 而本文的算法较为稳定, 能够收敛到接近最优解.

从图 10(b) 中可以看出, 随着初始种群规模的增大, 阶段 1 设备资源调度算法的对比算法 (蓝色和黄色折线) 的调度成本 *Cost* 值逐步减小, 两条折线的 *Cost* 平均值从 2 415 降低到 1 747. 而图 10 中红色和绿色折线所代表的本文算法 DRSA 的调度成本 *Cost* 值缓步下降, 两条折线的 *Cost* 平均数值从 1 500 降低到 1 495. 从数值上看, 无论阶段 2 使用哪种算法, 阶段 1 中本文算法 DRSA 与对比算法效果更好. 这是因为本文的算法受到种群规模影响较小, 在小规模种群下, 依然能收敛到接近最优解.

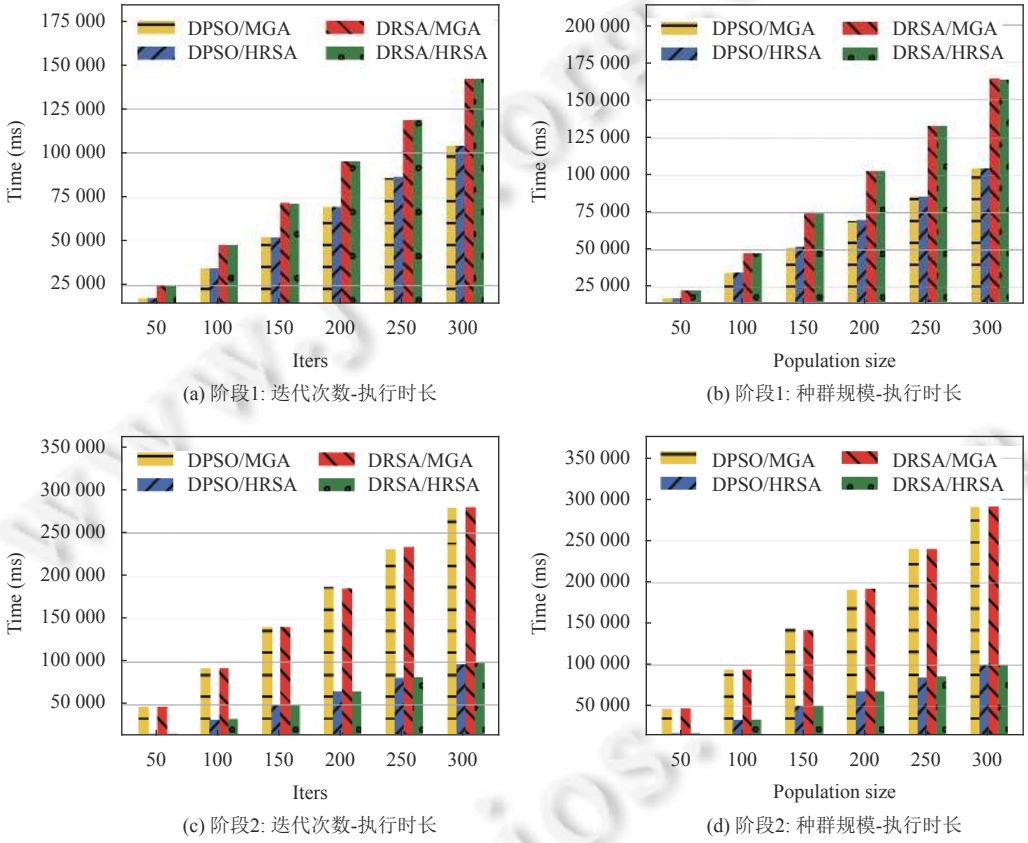
从图 10(c) 中可以看出, 随着迭代次数的增大, 阶段 2 人力资源调度算法的两个算法的调度成本 *Cost* 值均有所下降. 本文算法 HRSA (绿色和蓝色折线) 总体的调度成本 *Cost* 值好于对比算法 (红色和黄色折线), *Cost* 的平均数值. 本文算法 HRSA 的 *Cost* 平均值与对比算法相比数值更低.

从图 10(d) 中可以看出, 随着初始种群规模的增大, 阶段 2 人力资源调度算法的两个算法的调度成本 *Cost* 值均有所下降. 本文算法 HRSA (绿色和蓝色折线) 总体的调度成本 *Cost* 值好于对比算法 (红色和黄色折线).

● RQ2: 算法的效率

算法的效率实验目的是对两阶段算法的执行总时长进行评估. 具体而言, 首先保持的阶段 2 的种群规模和迭代次数不变, 分别改变阶段 1 的设备资源调度算法的迭代次数和种群规模并统计两阶段算法的运行总时长. 此时阶段 2 的两种算法迭代次数均设置为 100 次, 种群规模均设置为 100. 其次, 阶段 1 的种群规模和迭代次数不变, 分别改变阶段 2 的人力资源调度算法的迭代次数和种群规模并统计两阶段算法的运行总时长. 此时阶段 1 中两种算法迭代次数均设置为 100 次, 种群规模设置为 100. 实验运行 3 个周期, 保持每个周期输入调度框架的应用实例数量为 100 个. 设备资源服务点数量设置为 100, 工人数量设置为 20.

图 11(a) 展示了迭代次数对于阶段 1 设备资源调度算法运行时长的影响. 横坐标为阶段 1 算法的迭代次数. 图 11(b) 展示了初始种群规模对于阶段 1 设备资源调度算法运行时长的影响. 横坐标为阶段 1 算法的初始种群规模. 图 11(c) 展示了迭代次数对于阶段 2 人力资源调度算法运行时长的影响. 横坐标为阶段 2 算法的迭代次数. 图 11(d) 展示了初始种群规模对于阶段 2 人力资源调度算法运行时长的影响. 横坐标为阶段 2 算法的初始种群规模.



(a) 阶段1: 迭代次数-执行时长　(b) 阶段1: 种群规模-执行时长

(c) 阶段2: 迭代次数-执行时长　(d) 阶段2: 种群规模-执行时长

图 11　RQ2: 算法效率实验结果

从图 11(a) 中可以看出, 随着迭代次数的增大, 阶段 1 两个设备资源调度算法的运行时长也随之线性增大. 从图 11(b) 中可以看出, 随着初始种群规模的增大, 阶段 1 设备资源调度算法的对比算法的时长也随之线性增大从图 11(c) 中可以看出, 随着迭代次数的增大, 阶段 2 人力资源调度算法的运行时长也随之线性增大. 从图 11(d) 中可以看出, 随着初始种群规模的增大, 阶段 2 人力资源调度算法的运行时长也随之线性增大. 从结果来看, 无论是本文算法还是对比算法均受到迭代次数和种群规模的影响, 而且影响总体是呈线性. 这是因为算法的时间复杂度与迭代次数和种群规模线性相关. 而因为 DPSO 算法具有收敛快速的特点, 因此在阶段 1 中本文算法 DRSA 执行时长稍长于对比算法. 而在阶段 2 本文算法 HRSA 相比对比算法执行时长明显短于对比算法. 综合两阶段来看,

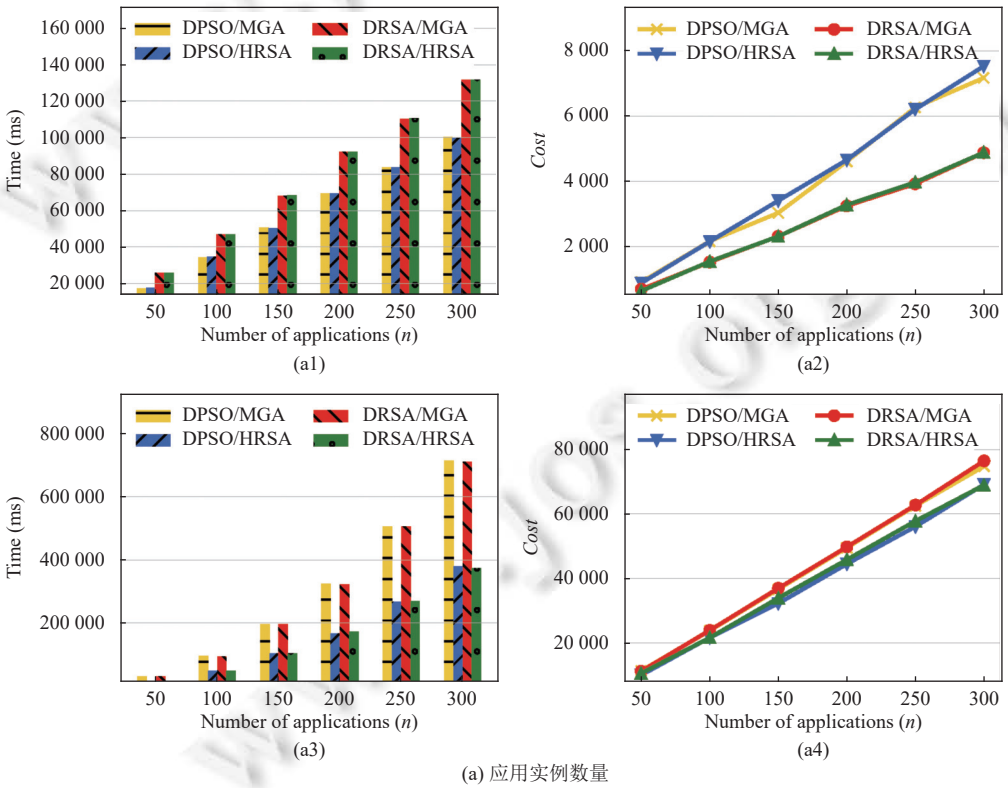
本文算法具有一定优势.

● RQ3: 其他参数对算法的影响

这部分实验是为了研究其他参数对算法执行总时长和调度总成本的影响, 分别从应用实例数量、设备资源服务点数量和工人数量 3 个方面进行评估. 在评估应用实例数量的影响时, 首先保持两阶段的种群规模和迭代次数不变, 同时保持设备资源服务点数量和工人数量不变, 改变在每个周期输入的应用实例数量. 具体地, 两阶段对比算法的迭代次数均设置为 100 次, 种群规模均设置为 100. 设备资源服务点数量设置为 100, 工人数量设置为 20, 应用实例数量作为变量, 其数值 $n \in \{50, 100, 150, 200, 250, 300\}$. 在评估设备资源服务点数量的影响时, 首先保持两阶段的种群规模和迭代次数不变, 同时保持应用实例数量和工人数量不变, 改变初始化时, 服务点的数量. 具体地, 两阶段对比算法的迭代次数均设置为 100 次, 种群规模均设置为 100. 设备应用实例数量为 100, 工人数量设置为 20, 设备资源服务点数量作为变量, 其数值 $n \in \{20, 40, 60, 80, 100\}$. 在评估工人数量的影响时, 首先保持两阶段的种群规模和迭代次数不变, 同时保持应用实例数量和设备资源服务点数量不变, 改变在每个周期输入的应用实例数量. 具体地, 两阶段对比算法的迭代次数均设置为 100 次, 种群规模均设置为 100. 应用实例数量为 100, 设备资源服务点数量设置为 100, 工人数量作为变量, 其数值 $n \in \{20, 40, 60, 80, 100\}$.

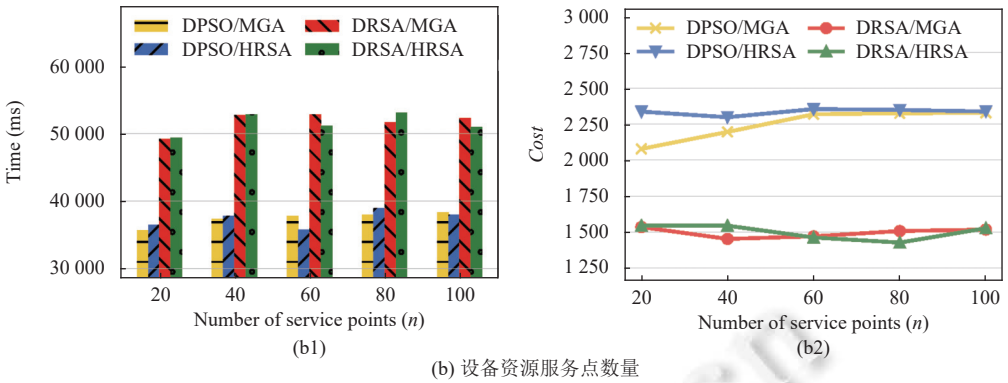
图 12(a) 展示了应用实例数量对于算法运行的影响. 横坐标为应用实例的数量. 图 12(a1)–图 12(a4) 这 4 个子图分别展示了设备资源调度阶段的算法运行时长 Time 对比, 设备资源调度阶段的算法调度成本 *Cost* 值对比, 人力资源调度阶段的算法执行时长 Time 对比和人力资源调度阶段的算法调度成本 *Cost* 值对比.

两个阶段共 4 个算法的运行时长均随着应用实例数量的增大而线性增大. 这是因为应用实例中包含了影响两个阶段的算法中搜索空间的设备子任务和工人子任务. 当任务数量增大时, 遗传算法中的染色体和离散粒子群优化中的位置编码的长度也会随之线性增加.

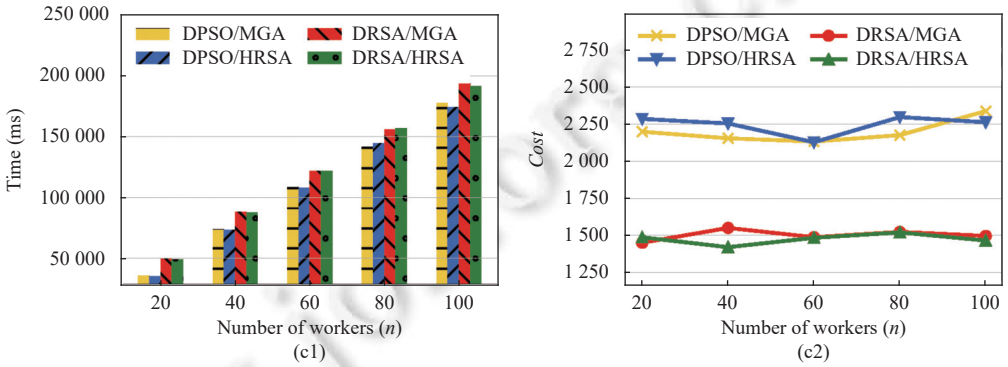


(a) 应用实例数量

图 12 RQ3: 外部环境对算法的影响



(b) 设备资源服务点数量



(c) 工人数量

图 12 RQ3: 外部环境对算法的影响 (续)

由图 12(a1) 和图 12(a2) 可以看出, 虽然设备资源调度阶段本文算法 DRSA 所花时间相对更长, 但最终调度成本 *Cost* 值更优. 其原因在于对比算法具有收敛快但容易陷入局部最优的特点. 由图 12(a3) 和图 12(a4) 可以看出, 在人力资源调度阶段本文算法 HRSA 和对比算法的最终调度成本 *Cost* 值稍优, 并且算法所花时间上有明显优势.

图 12(b) 展示了设备资源服务点数量对于阶段 1 设备资源调度算法运行的影响. 横坐标为设备资源服务点数量. 图 12(b1) 和图 12(b2) 两个子图分别展示了: 设备资源调度阶段的算法运行时长对比和设备资源调度阶段的算法调度成本 *Cost* 值对比.

在设备资源调度算法阶段, 这一阶段设备资源调度算法受到服务点数量的影响较小. 用时方面, 本文算法 DRSA 略微高于对比算法. 在 *Cost* 值方面, 本文算法优于对比算法.

图 12(c) 展示了工人数量对于阶段 2 人力资源调度算法运行的影响. 横坐标为工人数量. 图 12(c1) 和图 12(c2) 两个子图分别展示了人力资源调度阶段的算法运行时长 Time 对比和人力资源调度阶段的算法调度成本 *Cost* 值对比.

从图 12 中可以看出, 随着工人数量的增大, 对比算法的运行时长 Time 随之增大, 但本文算法 HRSA 用时相对稳定. 并且本文算法 HRSA 的调度成本 *Cost* 值相对更低.

5 相关工作

设备与人力资源协作任务的资源调度涉及的相关工作主要分为: 设备资源调度和人力资源调度问题.

5.1 设备资源调度问题

车间调度是面向设备资源调度的一种方式. 针对给定的工件和加工机器进行工件与加工机器的分配. 已有的调度算法分别针对车间调度模型和调度算法进行了大量研究. 在车间调度模型方面, Zhang 等人[6]提出了一种基

于博弈论的灵活车间调度动态优化模型, 根据其实时状态将任务分配给最优机器设备. Yuan 等人[7]提出了可重构装配线的多目标优化调度模型, 实现生产负载均衡, 并最小化延迟工作量. 在车间调度算法方面, Li 等人[11]提出了一种有效的混合算法, 该算法将遗传算法 (GA) 和禁忌搜索 (TS) 相结合, 以最小化完工时间. Wang 等人[12]提出了一种改进的粒子群优化 (PSO) 算法来解决动态车间调度问题. Defersha 等人[13]提出一种布谷鸟搜索和模拟退火 (CSA) 混合算法, 用于解决柔性车间调度中的连续优化问题. Chaouch 等人[14]使用 3 个版本改进版本的蚁群算法 (ACO) 的来处理动态车间调度问题. 除了使用启发式算法外, 基于强化学习的算法也被应用到车间调度问题, Chen 等人[15]提出了一种自学习遗传算法 (SLGA), 其中遗传算法 (GA) 作为基本优化方法, 并基于强化学习 (RL) 对其关键参数进行智能调整.

多阶段调度决策方法在灾后物品配送领域取得了一些研究成果. 例如, Kallaj 等人[16]使用了两阶段组成的随机规划方法来解决在不确定需求和捐献人数情况下的血液库存配送问题. 同样, Dukkanci 等人[17]也使用了两阶段组成的随机规划方法来解决在不确定需求和路线不确定情况下的救援物资配送问题. 然而, 上述研究中假设用于配送物资的无人机都是从固定仓库出发, 其灵活性不及本文中涉及的工人资源. Tirkolaee 等人[18]通过开发灰狼优化 (GWO) 和粒子群优化算法 (PSO), 解决供应链网络中的两级多产品位置分配路径问题. 然而, 上述工作涉及的场景是静态的调度场景, 与本文所探讨的动态 DHRC 任务不同.

边缘计算资源调度是另一种面向设备资源调度方式. 其针对给定的计算任务和边缘端设备进行计算任务与边缘端设备的分配, 将云端计算任务下发到分配完成的边缘端设备上. 胡晓敏等人[19]提出一种基于分层学习的粒子群算法, 优化每台移动设备对于移动设备的计算速度, 下载数据功耗, 数据卸载百分比和剩余网络带宽占这四个参数的取值, 更合理分配计算资源使得总能耗最小. Han 等人[20]针对任务卸载时通信成本无处不在且不对称的问题, 考虑了一种异构卸载模型, 提出一种基于半定松弛方法解决卸载问题. 一些研究也考虑到人的因素对于边缘计算资源的影响, 如 Deng 等人[21]通过考虑能耗、时延和价格, 提出了一种以用户为中心的任务卸载方案, 在满足用户个性化需求的约束下, 使用线性松弛改进的分枝定界算法和权值改进的粒子群优化算法来解决优化问题, 达到最小化时延、能耗和价格的加权成本. 这里的人主要指用户, 由于其不具备完成任务的能力, 因此用户不被视作人力资源. 绝大多数的设备资源调度算法均针对设备资源的分配, 对于和人力资源协同的调度算法研究较少.

5.2 人力资源调度问题

空间众包是面向人力资源的一种调度方式. 针对给定的一组任务集合和工人集合进行任务分配和工人的路径规划. 已有的调度算法分别针对任务分配和工人的路径规划进行了大量研究.

在任务分配算法上, Jiang 等人[22]考虑任务分配中工人质量和与任务距离两种策略进行任务分配. 文献 [23,24] 考虑了在任务分配中的公平性问题, 使工人平均报酬最大化的同时, 让工人间的报酬差异最小化. 文献 [25,26] 考虑了众包系统的隐私保护问题, 提出对工人和任务的位置等信息进行加密, 达到任务分配的同时有效保护隐私的目的. 范泽军等人[27]针对具有不同空间权限的众包工人, 提出多阶段任务分配方法, 将任务按照权限约束进行分解, 最后分配给具有相应权限的众包工人. Tran 等人[28]提出空间众包的实时任务分配框架, 在预算限制下达到任务分配数量最大化. Song 等人[29]考虑了不同技能的工人协作完成宏观任务的情况, 提出在线贪心算法, 实时为新出现的任务或工人计算最佳分配. 为了提高动态场景中任务调度的效果, 许多研究者提出基于工人和任务预测的众包任务分配方法. Zhao 等人[10]研究了基于预测的众包分配问题 (prediction task assignment, PTA), 使用时空循环神经网络 (ST-RNN) 预测每个工人在未来出现的位置并预测未来工人的可能行走的路线. Sun 等人[30]提出基于预测和自适应批处理的动态任务分配框架 (DTAF-PAB), 使用以门控循环单元 (gated recurrent unit, GRU) 为基本结构的深度神经网络预测未来任务数量.

在工人的路径规划算法上, Alabbadi 等人[31]在多目标方面进行了研究. 针对最大完成数、最小旅行成本和工人间任务量平衡 3 个优化目标, 提出了基于多目标粒子群优化和任务排序策略算法的 MOTSO 模型. 潘庆先等人[32]将时空众包中的任务分配问题建模为 (vehicle route planning, VRP) 路径规划问题, 提出基于自适应阈值的禁忌搜索算法, 在满足时空约束条件下获得最大任务分配数和工人和收益. Bouatouche 等人[33]将时空众包中的路径规划

问题建模为定向问题 (orienteering problem, OP), 提出了带禁忌搜索的贪心随机自适应搜索算法 (GRASP-Tabu), 找到优化路径使工人的收益最大化. Zheng 等人[34]构建了由工人、兴趣点和任务这 3 者组成的效用函数, 针对效用-代价比使用贪心算法进行优化调度, 同时设计了 SKT 算法减少局部最优. 崔俊云等人[35]研究了空间众包中固定终点的在线路径规划问题 (ORPP-FE), 提出基于粒子群优化的路径规划算法, 用于解决单工作者多任务情形下的动态路径规划问题. 沈彪等人[36]针对实时任务和动态工人的场景, 提出两阶段任务聚类的众包工人分配与路径规划方法, 并使用遗传算法进行求解. 然而其没有考虑任务之间的依赖关系.

绝大多数的人力资源调度算法均针对工人这一人力资源进行任务分配和规划, 对于和物理资源协同的调度算法研究较少.

6 总 结

针对人机物融合领域内设备与人力资源协作任务 (DHRC) 所具有的资源可选性、任务高频性、工人动态性的特点, 本文提出一种相应的资源优化调度方法, 通过两阶段的调度规划实现任务对设备资源以及人力资源的优化选择, 同时也为人力资源的行动规划路径. 基于本方法, 在模拟环境内的实验结果表明本文提出的基于 NSGA-II 和 DPSO 的资源调度方法可以得到有效的设备与人力资源调度决策, 在效率与效用性上基本均优于对比算法. 下一步工作将在此基础之上进一步考虑更多样的线下设备资源与人力资源的协作场景, 从而支持更复杂的协作场景的资源调度. 另外, 未来研究也将引入诸如任务难度、工人技能水平等因素来参与资源调度过程, 这些因素有助于更精确描述任务和资源间的关系, 从而获得更优、更全面的调度决策.

References:

[1] Mei H, Cao DG, Xie T. Ubiquitous operating system: Toward the blue ocean of human-cyber-physical ternary ubiquitous computing. Bulletin of Chinese Academy of Sciences, 2022, 37(1): 30–37 (in Chinese with English abstract). [doi: 10.16418/j.issn.1000-3045.20211117009]

[2] He FJ, Shen LW, Peng X. Resource choreography in cyber-physical-social systems: Representation, modeling and execution. IEEE Trans. on Services Computing, 2023, 16(1): 550–563. [doi: 10.1109/TSC.2021.3138637]

[3] Vavilapalli VK, Murthy AC, Douglas C, Agarwal S, Konar M, Evans R, Graves T, Lowe J, Shah H, Seth S, Saha B, Curino C, O'Malley O, Radia S, Reed B, Baldeschwieler E. Apache Hadoop YARN: Yet another resource negotiator. In: Proc. of the 4th Annual Symp. on Cloud Computing. Santa Clara: ACM, 2013. 5. [doi: 10.1145/2523616.2523633]

[4] Zhang Z, Li C, Tao YY, Yang RY, Tang H, Xu J. Fuxi: A fault-tolerant resource management and job scheduling system at Internet scale. Proc. of the VLDB Endowment, 2014, 7(13): 1393–1404. [doi: 10.14778/2733004.2733012]

[5] Hindman B, Konwinski A, Zaharia M, Ghodsi A, Joseph AD, Katz R, Shenker S, Stoica I. Mesos: A platform for fine-grained resource sharing in the data center. In: Proc. of the 8th USENIX Symp. on Networked Systems Design and Implementation. Boston: USENIX Association, 2011. 295–308.

[6] Zhang SC, Wong TN. Flexible job-shop scheduling/rescheduling in dynamic environment: A hybrid MAS/ACO approach. Int'l Journal of Production Research, 2017, 55(11): 3173–3196. [doi: 10.1080/00207543.2016.1267414]

[7] Yuan MH, Deng K, Chaovalitwongse WA, Cheng S. Multi-objective optimal scheduling of reconfigurable assembly line for cloud manufacturing. Optimization Methods and Software, 2017, 32(3): 581–593. [doi: 10.1080/10556788.2016.1230210]

[8] Tong YX, Zhou ZM, Zeng YX, Chen L, Shahabi C. Spatial crowdsourcing: A survey. The VLDB Journal, 2020, 29(1): 217–250. [doi: 10.1007/s00778-019-00568-7]

[9] Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. IEEE Trans. on Evolutionary Computation, 2014, 18(4): 577–601. [doi: 10.1109/TEVC.2013.2281535]

[10] Zhao Y, Zheng K, Cui Y, Su H, Zhu FD, Zhou XF. Predictive task assignment in spatial crowdsourcing: A data-driven approach. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering (ICDE). Dallas: IEEE, 2020. 13–24. [doi: 10.1109/ICDE48307.2020.00009]

[11] Li XY, Gao L. An effective hybrid genetic algorithm and Tabu search for flexible job shop scheduling problem. Int'l Journal of Production Economics, 2016, 174: 93–110. [doi: 10.1016/j.ijpe.2016.01.016]

[12] Wang Z, Zhang JH, Yang SX. An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with

random job arrivals. Swarm and Evolutionary Computation, 2019, 51: 100594. [doi: 10.1016/j.swevo.2019.100594]

[13] Defersha FM, Obimuyiwa D, Yimer AD. Mathematical model and simulated annealing algorithm for setup operator constrained flexible job shop scheduling problem. Computers & Industrial Engineering, 2022, 171: 108487. [doi: 10.1016/J.CIE.2022.108487]

[14] Chaouch I, Driss OB, Ghedira K. A modified ant colony optimization algorithm for the distributed job shop scheduling problem. Procedia Computer Science, 2017, 112: 296–305. [doi: 10.1016/j.procs.2017.08.267]

[15] Chen RH, Yang B, Li S, Wang SL. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Computers & Industrial Engineering, 2020, 149: 106778. [doi: 10.1016/j.cie.2020.106778]

[16] Kallaj MR, Kolaee MH, Al-E Hashem SMJM. Integrating bloodmobiles and drones in a post-disaster blood collection problem considering blood groups. Annals of Operations Research, 2023, 321(1–2): 783–811. [doi: 10.1007/s10479-022-04905-y]

[17] Dukkanci O, Koberstein A, Kara BY. Drones for relief logistics under uncertainty after an earthquake. European Journal of Operational Research, 2023, 310(1): 117–132. [doi: 10.1016/j.ejor.2023.02.038]

[18] Tirkolaee EB, Goli A, Mardani A. A novel two-echelon hierarchical location-allocation-routing optimization for green energy-efficient logistics systems. Annals of Operations Research, 2023, 324(1–2): 795–823. [doi: 10.1007/s10479-021-04363-y]

[19] Hu XM, Chen ZT, Li M. Level-based learning swarm optimization algorithm for resource scheduling in edge computing. Computer Engineering and Applications, 2022, 58(24): 107–115 (in Chinese with English abstract). [doi: 10.3778/j.issn.1002-8331.2110-0060]

[20] Han XX, Gao GC, Ning L, Wang Y, Zhang Y. A semidefinite relaxation approach for the offloading problem in edge computing. Computers & Electrical Engineering, 2022, 98: 107728. [doi: 10.1016/J.COMPELECENG.2022.107728]

[21] Deng XH, Sun ZH, Li D, Luo J, Wan SH. User-centric computation offloading for edge computing. IEEE Internet of Things Journal, 2021, 8(16): 12559–12568. [doi: 10.1109/JIOT.2021.3057694]

[22] Jiang Y, Cui LZ, Cao YM, Liu L, He W, Pan L, Zheng YQ, Li QZ. Spatial crowdsourcing task assignment based on the quality of workers. In: Proc. of the 3rd Int'l Conf. on Crowd Science and Engineering. Singapore: ACM, 2018. 28. [doi: 10.1145/3265689.3265717]

[23] Chen Z, Cheng P, Chen L, Lin XM, Shahabi C. Fair task assignment in spatial crowdsourcing. Proc. of the VLDB Endowment, 2020, 13(12): 2479–2492. [doi: 10.14778/3407790.3407839]

[24] Zhao Y, Zheng K, Guo JN, Yang B, Pedersen TB, Jensen CS. Fairness-aware task assignment in spatial crowdsourcing: Game-theoretic approaches. In: Proc. of the 37th IEEE Int'l Conf. on Data Engineering (ICDE). Chania: IEEE, 2021. 265–276. [doi: 10.1109/ICDE51399.2021.00030]

[25] Liu A, Wang WQ, Shang S, Li Q, Zhang XL. Efficient task assignment in spatial crowdsourcing with worker and task privacy protection. GeoInformatica, 2018, 22(2): 335–362. [doi: 10.1007/s10707-017-0305-2]

[26] Li MC, Wang JC, Zheng LB, Wu H, Cheng P, Chen L, Lin XM. Privacy-preserving batch-based task assignment in spatial crowdsourcing with untrusted server. In: Proc. of the 30th ACM Int'l Conf. on Information & Knowledge Management. Queensland: ACM, 2021. 947–956. [doi: 10.1145/3459637.3482288]

[27] Fan ZJ, Shen LW, Peng X, Zhao WY. Multi stage task allocation on constrained spatial crowdsourcing. Chinese Journal of Computers, 2019, 42(12): 2722–2741 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2019.02722]

[28] Tran L, To H, Fan LY, Shahabi C. A real-time framework for task assignment in hyperlocal spatial crowdsourcing. ACM Trans. on Intelligent Systems and Technology, 2018, 9(3): 37. [doi: 10.1145/3078853]

[29] Song TS, Xu K, Li JN, Li YM, Tong YX. Multi-skill aware task assignment in real-time spatial crowdsourcing. GeoInformatica, 2020, 24(1): 153–173. [doi: 10.1007/s10707-019-00351-4]

[30] Sun LJ, Yu XJ, Chen SC, Yan Y. A dynamic task assignment framework based on prediction and adaptive batching. In: Proc. of the 39th IEEE Int'l Performance Computing and Communications Conf. (IPCCC). Austin: IEEE, 2020. 1–8. [doi: 10.1109/IPCCC50635.2020.9391525]

[31] Alabbadi AA, Abulkhair MF. Multi-objective task scheduling optimization in spatial crowdsourcing. Algorithms, 2021, 14(3): 77. [doi: 10.3390/a14030077]

[32] Pan QX, Yin ZX, Dong HB, Gao XL, Tong XR. Spatiotemporal crowdsourcing task assignment algorithm based on Tabu search. CAAI Trans. on Intelligent Systems, 2020, 15(6): 1040–1048 (in Chinese with English abstract). [doi: 10.11992/tis.202006055]

[33] Bouatouche M, Belkadi K. GRASP-Tabu search algorithms for the route planning problem in spatial crowdsourcing. Int'l Journal of Applied Metaheuristic Computing, 2022, 13(1): 1–18. [doi: 10.4018/IJAMC.292502]

[34] Zheng BL, Huang CZ, Jensen CS, Chen L, Hung NQV, Liu GF, Li GH, Zheng K. Online trichromatic pickup and delivery scheduling in spatial crowdsourcing. In: Proc. of the 36th IEEE Int'l Conf. on Data Engineering (ICDE). Dallas: IEEE, 2020. 973–984. [doi: 10.1109/ICDE48307.2020.00089]

[35] Cui JY, Chen D, Yuan Y, Ma YL, Wang GR. Online route planning algorithm in spatial crowdsourcing. Journal of Tsinghua University

(Science and Technology), 2020, 60(8): 672–682 (in Chinese with English abstract). [doi: 10.16511/j.cnki.qhdxxb.2020.26.009]

[36] Shen B, Shen LW, Li Y. Dynamic task scheduling method for space crowdsourcing. Computer Science, 2022, 49(2): 231–240 (in Chinese with English abstract). [doi: 10.11896/jsjkx.210400249]

附中文参考文献:

[1] 梅宏, 曹东刚, 谢涛. 泛在操作系统: 面向人机物融合泛在计算的新蓝海. 中国科学院院刊, 2022, 37(1): 30–37. [doi: 10.16418/j.issn.1000-3045.20211117009]

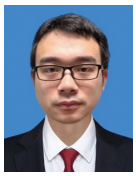
[19] 胡晓敏, 陈镇填, 李敏. 分层学习的边缘计算资源调度粒子群优化算法. 计算机工程与应用, 2022, 58(24): 107–115. [doi: 10.3778/j.issn.1002-8331.2110-0060]

[27] 范泽军, 沈立炜, 彭鑫, 赵文耘. 基于约束的空间众包多阶段任务分配. 计算机学报, 2019, 42(12): 2722–2741. [doi: 10.11897/SP.J.1016.2019.02722]

[32] 潘庆先, 殷增轩, 董红斌, 高照龙, 童向荣. 基于禁忌搜索的时空众包任务分配算法. 智能系统学报, 2020, 15(6): 1040–1048. [doi: 10.11992/tis.202006055]

[35] 崔俊云, 陈迪, 袁野, 马玉亮, 王国仁. 空间众包中在线路径规划算法. 清华大学学报 (自然科学版), 2020, 60(8): 672–682 [doi: 10.16511/j.cnki.qhdxxb.2020.26.009]

[36] 沈彪, 沈立炜, 李弋. 空间众包任务的路径动态调度方法. 计算机科学, 2022, 49(2): 231–240. [doi: 10.11896/jsjkx.210400249]



何飞佳(1992－), 男, 博士生, CCF 学生会员, 主要研究领域为泛在计算, 边缘计算, 人机物系统.

沈立炜(1982－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为移动应用程序开发和分析, 人机物融合系统.

宋育臣(1996－), 男, 硕士生, 主要研究领域为边缘计算, 人机物融合系统.

牛军钰(1967－), 女, 博士, 教授, 博士生导师, 主要研究领域为软件工程, 智能系统.

赵文耘(1964－), 男, 教授, 博士生导师, CCF 高级会员, 主要研究领域为软件工程, 智能化软件开发.