学习型变邻域搜索算法求解运输-装配协同优化问题

张腾飞[1]，胡蓉[1]，钱斌[1,2]，吕阳[1]
（1.昆明理工大学信息工程与自动化学院，云南 昆明 650500; 2. 昆明理工大学机电工程学院，云南 昆明 650500）

摘要：针对一类运输-装配协同优化问题，*建立其整数规划模型，提出一种融合分解策略的学习型变邻域搜索算法(Learning variable neighborhood search with decomposition strategy, LVNS_DS)对其求解. 首先，为降低问题的求解难度，设计一种分解策略将原问题分解为路径规划问题和装配线平衡问题. 其次，应用 LVNS 算法对两个子问题进行求解，然后通过合并子问题解可得原问题的完整解. 相比常规 VNS 算法, LVNS 算法依据邻域动作概率值来转换邻域结构，同时依据邻域动作产生的贡献来动态地更新其概率值，因此, LVNS 算法能以较大的概率值选择适于当前搜索阶段的邻域动作，从而易于找到子问题的优质解.* 通过不同规模算例的仿真实验，验证了运输-装配协同优化的有效性和 LVNS_DS 算法的有效性.

关键词: 协同优化；耦合性；装配线平衡；车辆路径优化；变邻域搜索；分解策略；
中图分类号：TP391.9 文献标志码：A 文献编号：1004-731X (2022)0132
DOI: 10.16182/j.issn1004731x.joss.22-0132

Learning Variable Neighborhood Search Algorithm for Transportation-assembly Collaborative Optimization Problem

Zhang Tengfei[1]，*Hu Rong*[1]，*Qian Bin*[1,2]，*Lv Yang*[1]

(1. School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, 650500, China;
2. School of Mechanical and Electrical Engineering, Kunming University of Science and Technology, Kunming, 650500,China;)

Abstract: *Aiming at transportation-assembly collaborative optimization problems, an integer programming model is established, and a learning variable neighborhood search with decomposition strategy (LVNS_DS) is proposed to solve it. Firstly, to reduce the difficulty of solving the problem, a decomposition strategy is designed to decompose the original problem into a path planning problem and assembly line balance problem. Secondly, the LVNS is used to solve the two subproblems, and the complete solution of the original problem can be obtained by merging subproblem solutions. Compared with the conventional VNS, the LVNS transforms the neighborhood structure according to the neighborhood action probability value, and dynamically updates its probability value according to the contribution of neighborhood action. Therefore, the LVNS algorithm can select the neighborhood action suitable for the current search stage with a large probability value, so it is easy to find the high-quality solution of the subproblem. Through the simulation experiments of different scale examples, the importance of transportation assembly collaborative optimization and the effectiveness of LVNS_DS has been verified.*

Key word: collaborative optimization; coupling; assembly line balance; vehicle routing optimization; variable neighborhood search; decomposition strategy;

收稿日期：2022-02-24 修回日期：2022-06-13
基金项目：国家自然科学基金项目(61963022,62173169)，云南省基础研究计划重点项目(202201AS070030)
第一作者：张腾飞(1997-)，男，硕士，研究方向为复杂系统建模与优化。E-mail：869959588@qq.com

引言

企业间协同发展是增强企业竞争力的重要手段[1]. 受全球新冠病毒影响, 全球性供应链逐渐缩短, 区域性供应链逐步加强. 区域性供应链通过准时化、智能化的数字平台使企业间价值传递更为高效、迅速, 有助于供应链企业实现协同发展[2-3]. 其中, 解决物流和生产协同优化问题是实现企业协同发展的具体手段. 与物流和生产协同优化问题相关的子问题通常具有 NP-hard 属性[4-5], 例如路径规划问题[6-8](Vehicle routing problem, VRP)、置换流水线问题[9](Permutation flow shop problem, PFSP)、简单装配线平衡问题[10-11](Simple assembly line balancing problem, SALBP)等. 针对这类问题的大规模实例, 精确算法难以在多项式时间内取得问题的最优解. 因此, 集成多类 NP-hard 子问题的物流和生产协同优化问题在问题建模和求解复杂度上远高于处理独立子问题的难度. 综上, 研究此类问题的合理建模方法与有效求解算法有助于实现企业间协同发展.

近年来, 一些学者研究了与物流和生产相关的协同优化问题. 例如, 邓超[12]等针对带配件批量运输的加工-运输-装配三阶段集成优化问题(Three-stage assembly integrated scheduling problem with job batch transportation, 3SAISP_JBT), 以最小化总完工时间为目标建立问题的排序模型, 采用融合规则的分布估计算法对其求解. 3SAISP_JBT 将装配阶段考虑为单机调度问题, 而实际装配环节通常带有复杂的工序优先约束、工序需求时间差别较大, 也未考虑路径规划对三者协同调度的影响. 近些年, AGV、智能仓储等技术迅速发展, 使物料供应的准时率不断提升, 为运输-装配协同优化提供技术支持. 周炳海[13]等针对准时化顺序供应的混流型装配线物料补给问题(Just-in-time material replenishment in mixed-model assembly problem, JITMR_MALP), 以工作站线边库存为优化目标建立问题的整数规划模型, 分别针对小规模问题、大规模问题应用反向动态规划算法与人工蜂群算法进行求解. 同样, JITMR_MALP 未考虑装配环节存在工序优先关系约束. Fathi[14]等针对厂内物流与装配问题, 以运输批次和装配厂库存成本为优化目标, 采用融合优先级规则的粒子群优化算法进行求解, 由于运输范围的限制, 并未考虑运输路径规划对运输成本的影响. 此外, Golz[15]等针对动态混流型装配线的物料补给问题(Part feeding problem at high-variant mixed-model assembly lines, PFP_HMAL), 以车辆运输成本和线边库存为目标, 基于车辆路径规划问题(Vehicle routing problem, VRP)建立该问题的整数规划模型. PFP_HMAL 将物料考虑为位于指定位置的单一仓库, 没有考虑企业间协同优化来降低库存. Rahman[16]等针对机器人装配线平衡及物料调度问题, 以装配节拍和总的拖延时间为优化目标建立问题的数学规划模型, 采用一种元启发式算法进行求解. 该模型考虑了工序约束、AGV 线路规划两者协同优化对物料供应和装配效率的影响, 受制于 AGV 运输线路、仓库数量及位置限制, 没有考虑 AGV 运输成本、线边库存与装配排序对运输-装配的整体影响.

上述文献以不同侧重研究了与物流和生产相关的协同优化问题, 然而, 多数文献尚未考虑装配阶段工序优先关系约束对运输-装配协同优化产生的影响, 因此, 本文研究了一类考虑工序优先关系约束的运输-装配协同优化问题(Transportation-assembly collaborative optimization problems, TACOP)并为其建立数学模型. TACOP 研究的是如何协调车辆运输方案与工序装配方案来最小化运输成本、库存水平、装配线节拍. 因此, TACOP 是同时考虑 VRP、SALBP 的两阶段协同优化问题. 其中, VRP 与 SALBP 均具有 NP-hard 属性, 在大规模问题上难以取得最优解, 而 TACOP 可行解数量近似等于 VRP 和 SALBP 可行解数量之积, 致使 TACOP 具有更强的 NP-hard 属性, 需要一种合理的分解策略对 TACOP 解空间进行有效划分. 此外, 虽然 TACOP 多个优化目标之间存在非支配关系, 但因问题存在明显的阶段性, 且关乎企业生产效率的生产节拍目标较其余目标拥有更高的优先级, 因此, 本

文以生产节拍为主要优化目标, 运输成本、库存水平的非支配解为次要优化目标, 建立了 TAOCP 的整数规划模型.

由于 TACOP 优化目标存在特殊性、问题本身具有 NP-hard 属性, 因此本文提出一种装配导向式分解策略(Assembly based decomposition strategy, ADS)来适应优化目标的特殊需求、同时缩小问题的解空间. ADS 将 TACOP 分解为包含多个 VRP 和一个 SALBP 的两阶段组合优化问题, 通过优先处理 SALBP 来获取包含优质装配解的集合, 其次, 基于这些优质装配解来处理 VRP 问题, 然后通过逆向计算得出车辆出发时间、配件抵达时间, 最后可得 TACOP 的各项优化指标. 为验证 ADS 有效性, 将其与运输导向式分解策略(Transportation based decomposition strategy, TDS)进行对比. 由于 TDS 不具备决策车辆发出时间的能力, 导致车辆抵达装配厂的时间存在较大波动, 又因装配过程存在工序优先关系约束, 易于出现因配件提前到达或到达不及而产生的库存堆积或库存不足等问题, 最终导致难以获取 SALBP 的优质解. 相比之下, 通过证明 SALBP 优质装配解存在非唯一性, 可使运输车辆与运输任务之间建立十分灵活的组合方案, 进而扩展了运输成本与库存水平的非支配解. 由于 ADS 不仅可以缩小 TACOP 的解空间, 而且其分解效果能够较好的适应优化目标之间的特殊关系, 因此, ADS 是一种适用于 TACOP 的分解策略.

变邻域搜索算法(Variable neighborhood search, VNS)是一种元启发式算法[17]. 与诸多智能算法相比, 由于 VNS 算法没有应用“群体”概念, 而是侧重于通过不同邻域结构之间的快速转换来迅速锁定解空间中的优质解, 所以 VNS 算法具有可变参数少、结构简单、收敛性好、易于定义离散问题邻域结构等优点, 已广泛应用于生产调度问题[17, 18]、物流调度问题[20]等众多组合优化问题. 由于 VRP 与 SALBP 同属具有 NP-hard 特性的整数规划问题, 该类问题又易于设计基于排序的搜索策略, 因此, 在 VNS 算法的基础上, 提出一种学习型变邻域搜索算法(Learning variable neighborhood search, LVNS)对 VRP 和 SALBP 两类整数规划问题进行求解, 并将求解原问题的算法称之为融合分解策略的学习型变邻域搜索算法(Learning variable neighborhood search with decomposition strategy, LVNS_DS). VNS 算法的关键在于设计适合问题性质的邻域结构、扰动算子执行条件及邻域结构执行顺序等因素[21]. 因此, 在 LVNS 算法中, 设计多种不同尺度的邻域动作, 将同类型全体邻域动作的集合定义为该类动作对应的邻域结构. 在处理大规模问题时, 由于邻域结构规模通常较大, 搜索单一邻域结构需要耗费较多计算时间资源, 且当算法陷入该邻域最优解之后, 执行剩余邻域的搜索任务将浪费额外的计算时间资源. 相比 VNS 算法, LVNS 算法利用不同类型邻域动作在不同邻域结构之间的转换搜索, 同时利用不同搜索时段邻域动作的表现对邻域动作的选择概率值进行实时地学习更新, 同时使用包含最优策略与随机策略的混合选择策略对学习后的邻域动作进行选择, 从而减小 VNS 算法易于陷入邻域结构内局部最优的不足, 提高算法的收敛性与求解质量. 综上, LVNS_DS 算法步骤可表述为, 首先利用 ADS 将 TACOP 分解为两类子问题, 然后, 利用 LVNS 算法对两类子问题进行求解, 以获取优质运输解与装配解. 其次, 将运输解与装配解合并为原问题解, 同时计算 TACOP 三个优化指标的值. 最后, 由迭代型算法的保优特性得到 TACOP 的优质解.

本文第一节建立运输-装配协同优化问题(TACOP)数学模型. 第二节分析 TACOP 耦合性, 确立装配导向式分解策略(ADS). 第三节介绍求解 TACOP 两类子问题的学习型变邻域搜索算法(LVNS)和求解原问题的 LVND_DS 算法. 第四节通过数值实验和算法对比验证运输-装配协同优化有效性和 LVNS_DS 算法求解 TACOP 的有效性.

1 问题模型

针对运输-装配协同优化问题(TACOP), 做如下假设: (1)装配厂需生产产品的工序优先关系约束已知; (2)装配工序所需配件来自不同配件厂; (3)配件

厂位置坐标已知; (4)所有环节均满足准时制原则; (5)单车辆至少能够满足单个取货点的需求.

TACOP 可简单描述为: 运输车自装配厂出发, 按照既定的路径, 各自前往任务集内配件厂取货, 然后返回装配厂. 装配线工作站线边配件按既定节拍被装配至线上. 产品按相同节拍在装配线移动, 并在末尾工作站形成成品. 在此之前, 装配线上的产品均为半成品. 求解 TACOP 是确定一种合适的车辆分配方案、车辆出发方案、车辆路径规划方案和装配排序方案, 在保证装配效率前提下, 得到运输成本、库存水平的非支配解集. 图 1 为该问题示意图. 模型中所用符号如表 1 所示.

表 1 数学符号及其表述

Tab.1 Mathematical symbols and expressions

符号	说明
n	装配工序总数
m	工作站总数
$\|S\|$	表示集合 S 内元素总数
S^{alb}	所有装配工序集合, $\left\|S^{alb}\right\|=n$
S_k^{car}	车辆 k 需要运输的工序集
S_j^{alb}	工作站 j 内装配工序集
$i\prec i'$	表示配件 i 先于 i' 装配, 亦表示解 i' 支配 i
$Pset$	表示工序约束矩阵, $Pset=\{(i,i')\mid i\prec i',i,i'\in S^{alb}\}$
m_i	配件 i 重量, $i=1,2,...,n$
t_i	配件 i 装配时间, $i=1,2,...,n$
CT	装配节拍
n^{car}	车辆数
C_{const}	车辆固定发车成本
C_{per}	每公里运输成本
T_j^{alb}	工作站 j 完工时间
$l_{i,i'}$	工厂 i 到工厂 i' 最短路程
$t_k^{car\_arr}$	车辆 k 抵达装配厂的时间
t_i^{arr}	配件 i 抵达装配厂的时间
t_i^{wait}	配件 i 未抵达, 而导致装配线等待时长
t_i^{begin}	配件 i 开始装配时间
t_i^{den}	配件 i 在装配厂被装配前的等待时长
$\overline{T_{re}}$	所有配件在装配厂的平均等待时间
TC	运输成本
Π^{ALB}	整条线上装配工序排序 $\Pi^{ALB}=\pi_1^{ALB},\pi_2^{ALB},...,\pi_n^{ALB}$
$\pi_{j,\bar{p}}^{alb}$	工作站 j 上最后被装配的工序, 即 $\bar{p}$ 表最后
Π_j^{alb}	工作站 j 上装配工序排序 $\Pi_j^{alb}=\pi_{j,1}^{alb},\pi_{j,2}^{alb},...,\pi_{j,\bar{p}}^{alb}$

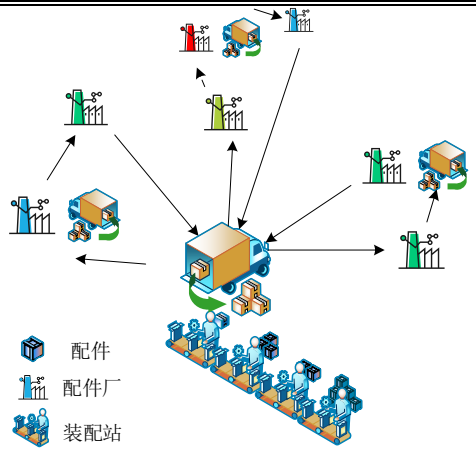

图 1 TACOP 模型示意图

Fig.1 Schematic diagram of TACOP model

1.1 TACOP 数学模型

(1)决策变量

$$z_{k,i,i'}=\begin{cases}1, \text{车辆}k\text{经工厂}i\text{ 到工厂}i'\\ 0, \text{车辆}k\text{未经工厂}i\text{ 到工厂}i'\end{cases} \tag{1}$$

$$w_{i,j,p}=\begin{cases}1, \text{配件}i\text{经工作站}j\text{ 以第}p\text{个装配}\\ 0, \text{配件}i\text{未经工作站}j\text{ 以第}p\text{个装配}\end{cases} \tag{2}$$

$$t_k^{car\_begin}\ge 0, k=1,2,...,n^{car} \tag{3}$$

(2)约束条件

$$\sum_{i=1}^{n}\sum_{i'=1}^{n}z_{k,i,i'}\le 1, k=1,2,...,n^{car} \tag{4}$$

$$\sum_{i\in S_k^{car}}m_i\le M, k=1,2,...,n^{car} \tag{5}$$

$$\sum_{i=1}^{n}\sum_{p=1}^{\bar{p}}w_{i,j,p}\times t_i\le CT, j=1,2,...,m \tag{6}$$

$$\sum_{j=1}^{m}\sum_{p=1}^{\bar{p}}w_{i,j,p}=1, i=1,2,...,n \tag{7}$$

$$\sum_{j=1}^{m}\sum_{p=1}^{\bar{p}}j\times w_{i,j,p}\le\sum_{j=1}^{m}\sum_{p=1}^{\bar{p}}j\times w_{i',j,p}\quad \forall(i,i')\in Pset \tag{8}$$

$$n_{\min}^{car}\le n^{car}\le n \tag{9}$$

(3)优化目标

$$\min TC=\min\left(C_{per}\times L+C_{const}\times n^{car}\right) \tag{10}$$

$$\min CT=\min\left(\max\left(T_1^{alb},T_2^{alb},...,T_m^{alb}\right)\right) \tag{11}$$

$$\min\overline{T_{re}}=\min\frac{\sum_{i=1}^{n}t_i^{den}}{n} \tag{12}$$

公式(1)用于决策车队中车辆按何种顺序完成运输任务, i 为0表示装配厂, i 其余取值表示配件厂. 公式(2)用于决策装配线如何安排各工作站以完成装配任务, 特别地, $w_{i,j,\bar{p}}=1$ 表示配件 i 在工作站 j 上以最后一个被装配, 即 $\bar{p}$ 表示最后一道装配.

公式(3)代表车辆的出发时刻. 约束式(4)表示车辆在发出后至多到达同一配件厂一次. 约束式(5)表示车辆运输零件重量不可超过车辆的载重上限. 约束式(6)表示工序分配至工作站时需满足装配线节拍. 约束式(7)表示每种工序仅能在某一工作站装配一次. 约束式(8)表示分配工序至工作站需满足工序约束矩阵 $Pset$. 约束式(9)表示车辆数目的取值约束. 其最小取值由公式(13)计算, 式中 nl 表示装配线并行条数. 由于单个配件不可分割, 所以最小取值可能无法构成可行解.

$$n_{\min}^{car}=\left\lceil\left(nl\times\left(\sum_{i\in S^{alb}}m_i\right)\right)\Big/M\right\rceil \tag{13}$$

TACOP 存在三个优化目标. 公式(10)计算运输成本目标, 其值与车辆数目、车辆任务集划分和车辆路径规划三者有关, L 表示车辆途径的总路程, 由公式(14)计算.

$$L=\sum_{k=1}^{n^{car}}\sum_{i'=0}^{n}\sum_{i=0}^{n}\left(z_{k,i,i'}\times l_{i,i'}\right) \tag{14}$$

公式(11)表示装配节拍目标，其值与配件抵达装配厂时间、配件装配排序两者有关, T_j^{alb} 表示 j 工作站完工时间, 由公式(15)计算.

$$T_j^{alb}=\sum_{p=1}^{\bar{p}}\sum_{i=1}^{n}\left(t_i+t_i^{wait}\right)\times w_{i,j,p} \tag{15}$$

其中 t_i^{wait} 表示因配件 i 未抵达而产生的等待时间, 因此, t_i^{wait} 取决于配件 i 的抵达时间和和上一个配件的完工时间, 由公式(16)计算.

$$t_i^{wait}=\begin{cases}0 & i=i\_0\\ 0 & i\neq i\_0, t_{i\_pre}^{finish}\geq t_i^{arr}\\ t_i^{arr}-t_{i\_pre}^{finish} & i\neq i\_0, t_{i\_pre}^{finish}<t_i^{arr}\end{cases} \tag{16}$$

$$i\_0=\sum_{i=1}^{n}i\times w_{i,1,1} \tag{17}$$

其中, i_0 表示首个被装配工序, 由公式(17)计算; $t_{i\_pre}^{finish}$, t_i^{arr} 分别表示配件 i 前序工序的完工时间和配件 i 的抵达时间, 分别由公式(18)和(19)计算. t_i^{finish} 表示配件 i 的完工时间, $t_{i\_pre}^{finish}$ 计算包含于 t_i^{finish} .

$$t_i^{finish}=t_i^{begin}+t_i \tag{18}$$

$$t_i^{arr}=\sum_{i'=1}^{n}\sum_{k=1}^{n^{car}}\left(t_k^{car\_arr}\times z_{k,i,i'}\right) \tag{19}$$

其中, t_i^{begin} 表示配件 i 开始装配的时间, 由公式(20)计算, $\hat{j}$ 表示配件 i 被装配的工作站, 由公式(21)计算. $t_k^{car\_arr}$ 表示车辆 k 抵达装配厂的时间, $t_k^{car\_arr}$ 由公式(23)计算, $speed$ 表平均车速.

$$t_i^{begin}=\begin{cases}t_i^{arr} & i=i\_0\\ \max\left(t_{i\_pre}^{finish},t_i^{arr}\right) & i\neq i\_0, t^{sapce}\geq t_i\\ \max\left(CT\times\hat{j},t_i^{arr}\right) & i\neq i\_0, t^{sapce}<t_i\end{cases} \tag{20}$$

$$\hat{j}=\sum_{p=1}^{\bar{p}}\sum_{j=1}^{m}\left(j\times w_{i,j,p}\right) \tag{21}$$

式(20)中 t^{sapce} 表示在装配节拍 CT 下工作站剩余时间容量, 由公式(22)计算.

$$t^{sapce}=CT\times(\hat{j}-1)+t_{i\_0}^{arr}-t_{i\_pre}^{finish} \tag{22}$$

$$t_k^{car\_arr}=t_k^{car\_begin}+\sum_{i'=0}^{n}\sum_{i=0}^{n}\left(z_{k,i,i'}\times l_{i,i'}\right)\Big/speed \tag{23}$$

公式(12)用于计算库存水平目标, 该目标通过配件在装配厂的平均滞留时间来衡量, t_i^{den} 表示配件 i 在装配厂被装配前的等待时间, 由公式(24)计算.

$$t_i^{den}=\begin{cases}0 & i=i\_0\\ 0 & i\neq i\_0, t_{i\_pre}^{finish}\leq t_i^{arr}\\ t_{i\_pre}^{finish}-t_i^{arr} & i\neq i\_0, t_{i\_pre}^{finish}>t_i^{arr}\end{cases} \tag{24}$$

在实际生产中, 装配效率相比运输成本、库存水平拥有更高的优先级. 因此, 本模型中将装配节拍设置为主要优化目标, 其余两个目标的非支配解设置为次要目标.

1.2 问题示例

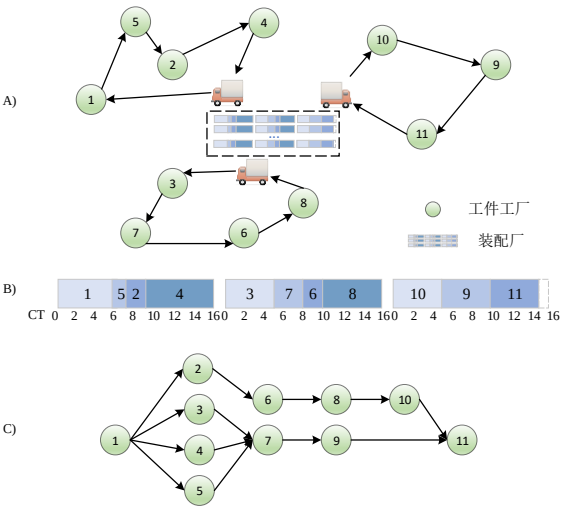

图 2 A) 以 P9 规模产品为例的 TACOP 模型求解示例
B)装配过程排序 C)工序优先关系有向图
Fig.2 A) Solution example of TACOP model (P9)
B) Assembly process sequencing C) Directed graph of operation priority relationship

图 2 为 TACOP 示例. 该例中, 一类拥有 9 道装配工序的产品需在装配厂完成装配. 每道装配工序对应一种配件. 图中装配厂不生产、储备配件, 配件在需要的时候由配件厂运输至装配厂进行装配. 在考虑配件厂位置、装配工序约束、车辆等因素后, 计算出如图所示运输-装配协同优化方案. 该方案较为充分地考虑了配件厂位置因素与配件装配顺序之间的关系, 减小了配件在装配厂的囤积时间, 即缩小了库存成本, 同时, 也取得了较好的运输、装配方案, 从而实现物流-装配协同优化.

2 问题的性质与分析

耦合性多用于软件度量, 是指程序中各模块之间的信息、参数依赖程度[22]. 运输-装配协同优化问题(TACOP)是一类两阶段组合优化问题. 本文用耦合性衡量两阶段的依赖程度、阶段内元素之间的依赖程度. 因此, 本节按阶段间耦合性(Coupling between stages, CBS)与阶段内部耦合性(Coupling within stages, CWS)对 TACOP 决策空间复杂度进行分析. 在问题分析基础上, 提出此问题分解策略, 以降低该问题的求解难度.

2.1 阶段间耦合性分析

在运输-装配协同优化问题(TACOP)中, 阶段间耦合性用于度量运输、装配两阶段间的关联性. 耦合性较弱表明运输阶段、装配阶段的独立性较强. 反之, 独立性较弱.

阶段间耦合性体现于装配线站边库存. 当装配线线边库存充足时, 耦合性较弱. 当缩减线边库存后, 两阶段耦合性逐步增强, 各阶段优化目标开始相互影响. 由于需要同时解决运输成本、库存水平与装配效率的平衡问题, 所以阶段间具有耦合性的协同优化问题比单阶段问题的决策空间复杂度大得多. 在实际问题中, 因为库存水平直接影响企业的成本开支, 所以优化企业成本必然需要面临协同优化中的耦合性. 为降低 TACOP 的求解难度, 提出装配导向式分解策略(Assembly based decomposition strategy, ADS), 并将其与运输导向式分解策略(Transportation based decomposition strategy, TDS)做对比, 从决策空间复杂度与实际需求两个角度分析、验证了 ADS 是一种合理、有效的分解策略.

2.1.1 以 TDS 对 TACOP 分析

运输导向式分解策略(TDS)首先优化运输环节, 当一个运输解确定后, 再依据该解的配件抵达时间对装配环节进行优化, 例如“先到先装配”规则[12], 然后可得该运输解对应的装配解, 最后由两阶段解共同构成 TACOP 的完整解. TDS 对优化指标和解空间的影响具体分析如下:

在给定装配任务后, n^{car} 辆运输车从装配厂同时发, 即决策变量 $t_k^{car\_begin}=0\left(k=\left(1,2,...,n^{car}\right)\right)$, 依次前往任务集 S_k^{car} 内配件厂取货, 然后返回装配厂. 由于取货环节满足准时制原则, 所以车辆返回装配厂的时间 $t_k^{car-arr}\left(k=\left(1,2,...,n^{car}\right)\right)$ 可以被计算, 因此, 可确定各配件抵达装配厂的具体时间 t_i^{arr}. 基于“先到先装配”规则[12], 在满足工序优先关系约束和节拍约束的条件下, 将抵达配件厂的配件分配至装配线工作站, 应用智能算法的迭代特性, 不断压缩节拍约束以得到 SALBP 的优质解. 最后, 利用配件抵达装配厂时间与被装配时间之差, 计算配件在装配厂的滞留时间、因配件未抵达而造成的装配线等待时间. 由模型中, 公式(16)和(24)可知, 在该分解策略下, 由于运输车返回装配厂时间不一致, 导致装配线站边库存出现不同程度“间断”或“阻塞”, 对企业的生产效率产生直接影响.

$$A=\prod_{k=1}^{n^{car}}\left(\left|S_k^{car}\right|!\right) \tag{25}$$

$$B=\prod_{j=1}^{m}\left(\left|S_j^{alb}\right|!\right) \tag{26}$$

此外, 依据公式(25)(26)对该分解策略下问题决策空间复杂度进行分析. 公式(25)中 A 近似表示运输阶段的问题决策空间复杂度. 公式(26)中 B 近似表示装配阶段问题决策空间复杂度. TDS 策略下运输-装配协同优化问题的决策空间复杂度近似表示为 $A\times B$.

2.1.2 以 ADS 对 TACOP 分析

ADS 是一种以装配需求为导向的逆向分解策略. 如图 3 所示. 该策略将 TACOP 分解为一个

SALBP 和多个 VRP，并首先对 SALBP 进行求解，然后对 VRP 进行求解，最后再将两阶段解组合为原问题解.

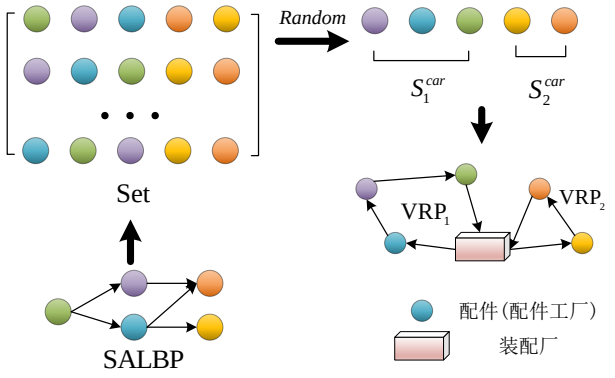

图 3 ADS 分解策略下 TACOP 求解流程
Fig.3 TACOP solving process under ADS decomposition strategy

由下述性质 1 可知，相同优质节拍下 SALBP 通常存在多个优质解. 算法 1 能够利用单个优质装配解扩展计算出包含多个优质装配解的集合. 在此基础上，从装配解集合中随机选取装配解，以算法 2 所述启发式方法将装配解包含的配件(配件厂)分配至 n^{car} 辆运输车，从而得到车辆的运输任务集. 其中，$\overline{M}$ 表示车辆平均载重，由公式(28)计算. 设置 $\overline{M}$ 目的是均衡各车辆任务集大小，其原因在于，均衡各车辆载重不仅能够降低 n^{car} 个 VRP 子问题的解空间，而且能够平衡各时段工作站的库存水平，从而避免某一工作站存在较大的库存堆积. 得到运输任务集后，将每个任务集看做一个独立的 VRP 子问题，因此可以同时对 n^{car} 个子问题进行求解. 在装配排序、车辆路径排序确定后，由于 ADS 策略下 $t_i^{wait}=0\left(i=(1,2,...,n)\right)$，所以可反向推导出各车辆的出发时间 $t_k^{car\_begin}$. 最后，计算运输成本、库存水平，实现算法的一次迭代过程. 在每次迭代中，算法能获取一组问题的解. 该解的评价值由装配节拍、库存水平、运输成本构成. 在迭代中，首先对比主要目标装配节拍，其次，对比次要目标库存水平、运输成本构成的非支配解. 在算法终止时能够得到一组满足模型中优化目标的优质解集. 综上，该装配导向式分解策略(ADS)不仅能够保证企业生产效率，而且可以平衡各车辆间运输任务集、能够决策车辆出发时间，所以该策略能够较好的适应运输-装配协同调度问题.

性质 1：在 SALBP 中，若装配线处于最优节拍下，且某工作站中存在优先关系互不支配又相邻的工序，则该装配线最优节拍下的最优装配解不唯一.

证明：设装配线达到最优节拍(或近似最优节拍) CT^*. $S_1^{alb},S_2^{alb},...,S_m^{alb}$ 为 m 台工作站包含的工序集. $\Pi_j^{alb}=\pi_{j,1}^{alb},\pi_{j,2}^{alb},...,\pi_{j,\bar{p}}^{alb}\left(j=(1,2,...,m)\right)$ 为 m 台工作站工序排序. $\pi_{j,\bar{p}}^{alb}$ 表示第 j 台工作站最后一项装配工序. 若工序 $\pi_{j,i}^{alb}$ 与工序 $\pi_{j,i+1}^{alb}$ 满足 $\pi_{j,i}^{alb}\not\prec\pi_{j,i+1}^{alb}$ 且 $\pi_{j,i+1}^{alb}\not\prec\pi_{j,i}^{alb}$，则交换 $\pi_{j,i}^{alb}$ 和 $\pi_{j,i+1}^{alb}$ 可得到新解 $\Pi_j^{alb'}=\pi_{j,1}^{alb'},\pi_{j,2}^{alb'},...,\pi_{j,\bar{p}}^{alb'}\left(j=(1,2,...,m)\right)$. 此时，$j$ 工作站完工时间表示为 $T_j^{alb'}$. 由式(27)可推导 $T_j^{alb'}=T_j^{alb}$，又因装配节拍由完工时间最大的工作站决定，所以 $\Pi_j^{alb'}\left(j=(1,2,...,m)\right)$ 也是最优解. 因此，最优节拍下的最优装配解不唯一. **证毕**.

$$T_j^{alb'}=\sum_{i\in S_j^{alb}}t_i'=\sum_{i\in S_j^{alb}}t_i=T_j^{alb} \tag{27}$$

$$\overline{M}=\frac{\sum_{i\in S^{alb}}m_i}{n^{car}} \tag{28}$$

$$\begin{aligned}&\prod_{k=1}^{n^{car}}\left(\left|S_k^{car\_ADS}\right|!\right)\\&=\left(\left|S_1^{car\_ADS}\right|!\right)\times\left(\left|S_2^{car\_ADS}\right|!\right)\times...\times\left(\left|S_{n^{car}}^{car\_ADS}\right|!\right)\\&=\left(a_1!\right)\times\left(a_2!\right)\times...\times\left(a_{n^{car}}!\right)\\&=n!\end{aligned} \tag{29}$$

$$\begin{aligned}A\times B&=\prod_{k=1}^{n^{car}}\left(\left|S_k^{car}\right|!\right)\times B\\&=\left(\left|S_1^{car}\right|!\right)\times\left(\left|S_2^{car}\right|!\right)\times...\times\left(\left|S_{n^{car}}^{car}\right|!\right)\times B\\&>\left(\left|S_k^{car}\right|!\right)\times B=\left(\left|C_n^1+C_n^1+...+C_n^n\right|!\right)\times B\\&=\left(\left|2^n-1\right|!\right)\times B\\&>n!\times B\end{aligned} \tag{30}$$

此外，依据公式(29)、(30)对该分解策略下的问题决策空间复杂度进行分析. 装配环节决策空间复杂度同 TDS 下复杂度相同，近似用 B 表示. 运输环节的决策空间复杂度远小于 $n!$，该值由式(29)推导得出，$S_k^{car\_ADS}$ 表示 ADS 策略分配得到的运输任务集，运输任务数是固定值，进而可用 a 表示. 因此，在该分解策略下，TACOP 的决策空间复杂度远小于 $n!\times B$. 相比 $A\times B$，ADS 策略下的决策空间复杂度明显缩小，比较过程如式(30). 综上，从问题求解复

杂度层面比较, ADS 较 TDS 拥有更小的决策空间, 进而降低求解难度.

算法 1 优质装配排序的扩展方法

输入: 优质装配解
输出: 优质装配解集合 *Set*
$Set \leftarrow \Pi_j^{alb}\left(j=(1,2,...,m)\right)$, $count \leftarrow 0$
Repeat
  $count \leftarrow count+1$
  $\Pi_j^{alb'}\left(j=(1,2,...,m)\right) \leftarrow$随机取 *Set* 中一个解
  $J \leftarrow \mathrm{random}(1,\mathrm{m})$
  $\Pi_j^{alb''}\left(j=(1,2,...,m)\right) \leftarrow$ 交换 $\Pi_J^{alb'}$ 任意相邻工序
  If $\Pi_j^{alb''}\left(j=(1,2,...,m)\right)$ 不违背 *Pset* **then**
    $Set \leftarrow Set+\left\{\Pi_j^{alb''}\left(j=(1,2,...,m)\right)\right\}$
  End if
Until $count=n\times 10$

算法 2 获取车辆任务集 $S_k^{car}\left(k=(1,2,...,n^{car})\right)$

输入: $n^{car},\overline{M},n,\Pi^{ALB}=\pi_1^{ALB},\pi_2^{ALB},...,\pi_n^{ALB}$
输出: $S_k^{car}\left(k=(1,2,...,n^{car})\right)$
$i \leftarrow 0$, $M_k \leftarrow 0\left(k=(1,2,...,n^{car})\right)$
$k \leftarrow 1$
Repeat
  $i \leftarrow i+1$
  If $m_i+M_k>\overline{M}$ **then**
    $k \leftarrow k+1$
  End if
  $M_k \leftarrow M_k+m_i$, $S_k^{car} \leftarrow S_k^{car}+\left\{\pi_i^{ALB}\right\}$
Until $i=n$
For $k=1$ **to** n^{car} **do**
  If $M_k=0$ **then**
    $n^{car} \leftarrow k-1$, **Break**
  End if
End for

2.2 阶段内耦合性分析

阶段内部耦合性(CWS)包括子问题 SALBP 耦合性和多个 VRP 之间的耦合性. 其中, 多个 VRP 之间的耦合性分析已包含于 2.1 节中. 本节主要针对 SALBP 耦合性进行分析.

2.2.1 装配环节耦合性分析

SALBP 耦合性体现于工序间优先关系约束, 即工序之间的约束越多, SALBP 问题耦合性越强. 如图 4a 所示.

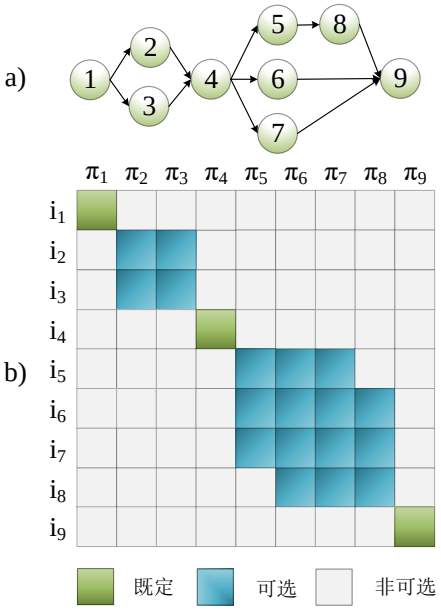

图 4 装配约束有向图与其工序矩阵(P9)
Fig.4 Assembly constrained directed graph and process matrix(P9)

一件拥有 n 项工序的产品, 在不考虑工序优先关系约束下, 其装配排序的可能性为 $n!$ 种. 然而, 考虑该约束来构建装配解可以大大降低决策空间复杂度, 具体分析如下:

设工序编号为 $i=(1,2,...,n)$, 装配排序为 $\Pi^{ALB}=\pi_1^{ALB},\pi_2^{ALB},...,\pi_p^{ALB},...,\pi_n^{ALB}$, $D_{n\times n}$ 为连接工序编号与装配排序变量的决策矩阵. 三者关系可表示为式(31). 令 $D_{n\times n}$ 内元素 $d_{i,p}$ 取值为 0 或 1, $d_{i,p}=1$ 表示工序 i 允许在第 p 个被装配, 即 $\pi_p^{ALB} \leftarrow i$. $D_{n\times n}$ 所有元素的取值由式(32)确定. 式中, τ 所在区间表示 $D_{n\times n}$ 中 i 行取值为 1 的元素, i 行其余非此区间内元素取值为 0, n_i^{pre} 与 n_i^{pro} 分别表示工序 i 前项约束工序的数目和后项约束工序的数目, 该值由工序优先关系计算得到. 应用上述理论对 P9 算例装配顺序进行分析. P9 算例 $D_{9\times 9}$ 由图 4b 可视化呈现. 绿色元素取值为 1, 表示该装配次序仅一种工序可供选择. 蓝色元素为可选元素, 即 0、1 取值不

确定. 白色元素取值均为 0, 表示某些工序不可能出现在某些装配次序. 由此可计算 P9 算例装配排序解空间为 $48(2!\times 4!)$ 种, 显然, 该值远小于 362880(9!)种. 因此, 基于工序优先关系约束构建的工序矩阵来生成装配解可有效降低 SALBP 解空间大小.

$$\left(\pi_1^{ALB},\pi_2^{ALB},...,\pi_n^{ALB}\right)=D_{n\times n}\times(1,2,...,n) \quad (31)$$

$$n_i^{pre}+1<\tau<n-n_i^{pro},i=(1,2,...,n) \quad (32)$$

3 LVNS_DS 算法求解 TACOP

LVNS_DS 算法由 ADS 和 LVNS 算法共同构成. 其中, ADS 已于 2.1.2 节介绍. LVNS 算法是一种基于 VNS 算法框架改进的元启发式算法. 相比 VNS 算法, LVNS 算法利用邻域动作转换来替换邻域结构遍历, 且该转换过程具备邻域动作学习能力与邻域动作选择能力. 本节首先介绍 VNS 算法框架, 其次介绍整数编码下的邻域动作类型及定义、邻域选择函数和邻域学习函数. 然后介绍如何利用 LVNS 算法求解 TACOP 两类子问题, 最后介绍 LVNS_DS 算法求解 TACOP 具体流程.

3.1 标准 VNS 算法框架

VNS 算法框架主要包括扰动算子和变邻域下降(Variable neighborhood descent, VND)算子. 算法 3 表示解决最小化问题的 VNS 算法框架. 其中, $\{N_k\,|k=1,2,...,k_{\max}\}$ 为一系列邻域结构. 在 VNS 算法中, VND 算子是依据预设的顺序依次遍历邻域结构, 以邻域结构内的局部最优解对历史最优解进行更新. 在整数规划模型中, 获取邻域结构局部最优解通常需要耗费相当多的计算时间资源, 原因在于, 简单定义下的邻域结构通常对应一个相当大的搜索空间. 例如, 针对拥有 20 个元素的编码排序, 获取单次插入算子的局部最优解需要处理 $2.43\times10^{17}\,(2\times19!)$ 个排序, 且通常需要对每个排序进行解码与评价. 因此, 以邻域结构局部最优解的方式寻找全局最优解存在搜索效率不高、搜索资源浪费的不足.

算法 3 解决最小化问题的 VNS 算法框架

```
输入: 邻域结构、终止条件和参数设置
输出: 最优或近似最优解 x_best
生成初始编码及初始解 x_0 ; x_best ← x_0
Repeat
    x' ← 由邻域结构 {N_k | k = 1,2,...,k_max} 生成可行解
    k ← 1
    Repeat
        x'_best ← x' 邻域结构 N_k 的局部最优解
        If x'_best < x' then
            x' ← x'_best , k ← 1
        Else
            k ← k + 1
        End if
    Until k = k_max
    If x' < x_best then
        x_best ← x'
Until 满足终止条件
```

3.2 邻域动作类型及定义

邻域动作(Neighborhood action, NA)是改变解邻域结构的具体方法. 由于子问题 SALBP 和 VRP 的解位于离散空间, 通常对此类问题进行整数编码, 因此, 设 $\pi_1^{subp},\pi_2^{subp},...,\pi_n^{subp}$ 为子问题(Subproblem)的通用编码. 针对该编码方式, 定义了四种不同尺度的邻域动作. 图 5 为邻域动作示意图. 为方便表述, 定义了与邻域动作相关的符号, 如表 2 所示.

NA_1: 从编码排序中随机取一位元素 π_i^{subp}, 将其与 π_{i+1}^{subp} 交换位置.

NA_2: 从编码排序中随机取两位元素 π_i^{subp} 与 $\pi_{i'}^{subp}$, 然后交换两元素的位置.

NA_3: 从编码排序中随机取两位元素 π_i^{subp} 与 $\pi_{i'}^{subp}$, 将元素 π_i^{subp} 插入 $\pi_{i'}^{subp}$ 之后.

NA_4: 从编码排序中随机取两位元素 π_i^{subp} 与 $\pi_{i'}^{subp}$, 将两元素之间的元素(包含两元素)以逆序方式重排.

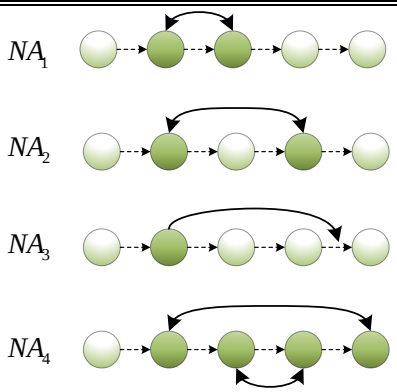

图 5 邻域动作示意图
Fig.5 Neighborhood action diagram

表 2 邻域动作及相关符号
Tab.2 Neighborhood actions and related symbols

动作名称	相邻逆序	交换	插入	局部逆序
编号	NA_1	NA_2	NA_3	NA_4
概率值	P_1	P_2	P_3	P_4

3.3 邻域动作选择及更新函数

在标准 VNS 算法中，邻域结构转换关系是预先设置的，通常由随机式或启发式规则对邻域结构的顺序进行排序. 然后，算法按照既定顺序逐一地对邻域结构进行搜索. 由于问题类型、问题规模和不同搜索阶段对邻域动作的需求存在差异，所以按照搜索需求选择合适的邻域结构十分必要. 因此，提出一种包含学习型邻域动作选择方式的 LVNS 算法. LVNS 算法邻域动作选择过程由选择函数决定，该函数如算法 4.

算法 4 NA 的选择函数

输入: 邻域动作选择概率 $P_k(k=1,2,3,4)$
输出: NA_s
$a \leftarrow random(0,1)$
If $0<a<0.7$ **then**
  $P_{\max} \leftarrow \max\left(P_k(k=1,2,3,4)\right)$
  $NA_s \leftarrow NA_{\max}$
Else
  $b \leftarrow random(1,2,3,4)$
  $NA_s \leftarrow NA_b$
End if

算法中首先赋予邻域动作相同初始值，该值由式(33)计算，n_A 表示邻域动作种类数. 在邻域动作选择过程中，不仅采用了最优策略，同时采用一定的随机策略. 邻域动作概率值由学习函数依据式(34)、(35)更新，该过程如算法 5. 其中，α 表示奖励系数，β 表示惩罚系数. 奖惩或惩罚由每种邻域动作各自表现决定.

算法 5 P_k 的更新函数

输入: 执行 NA_s 后的解 x ,最优解 x_{best} , 参数设置
输出: $P_k(k=1,2,3,4)$
$k \leftarrow s$
If $x < x_{best}$ **then**
  $\omega \leftarrow \alpha$
  P_k 由式(34)递增
Else
  $\omega \leftarrow -\beta$
  P_k 由式(34)递减
End if
由式(35)对 $P_k(k=1,2,3,4)$ 做归一化处理

$$P_k^0 = \frac{1}{n\_A}, k=(1,2,3,4) \tag{33}$$

$$P_k = \frac{(1+\omega)\times P_k}{\sum_{k=1}^{n\_A} P_k}, k=(1,2,3,4) \tag{34}$$

$$\omega = \begin{cases} \alpha, & NA_k \text{执行后评价值下降} \\ -\beta, & NA_k \text{执行后评价值未下降} \end{cases} \tag{35}$$

3.4 LVNS 算法求解 TACOP 子问题

3.4.1 子问题编码方法

SALBP 和 VRP 是 TACOP 的两类子问题. 由于两问题在结构上存在相似性，所以采用相同的编码方法. 具体地，采用一维排序的方式表示 SALBP 中工件装配排序和 VRP 中车辆途径的工厂排序. 不同点在于，SALBP 中编码排序不能随机排列，其排列方式受 *Pset* 约束，而 VRP 中编码排序不受工厂抵达顺序的约束. 本文用 $\pi_1^{ALB},\pi_2^{ALB}...,\pi_i^{ALB},...,\pi_n^{ALB}$ 表示 SALBP 编码排序，用 $\pi_1^{VRP},\pi_2^{VRP}...,\pi_i^{VRP},...,\pi_{n\_p}^{VRP}$ 表示

VRP 中编码排序，n_p 等于车辆需抵达工厂的数目.

3.4.2 子问题解码和评价方法

算法 6 SALBP 解码方法

输入: $\Pi^{ALB}=\pi_1^{ALB},\pi_2^{ALB},...,\pi_n^{ALB},m,CT$
输出: 装配解
$count\_n\leftarrow 0$，$j\leftarrow 1, count\_i\leftarrow 1$
Repeat
  $count\_n\leftarrow count\_n+1$
  $i\leftarrow \pi_{count\_n}^{ALB}$
  If $T_j^{alb}+t_i\le CT$ **or** $j=m$ **then**
    $T_j^{alb}\leftarrow T_j^{alb}+t_i$，$\pi_{j,count\_i}^{alb}\leftarrow i$
  Else
    $j\leftarrow j+1$，$count\_i=1$
    $T_j^{alb}\leftarrow T_j^{alb}+t_i$，$\pi_{j,count\_i}^{alb}\leftarrow i$
  End if
  $count\_i\leftarrow count\_i+1$
Until $count\_n=n$

算法 7 VRP 解码方法

输入: $\Pi^{VRP}=\pi_1^{VRP},\pi_2^{VRP},...,\pi_{n\_p}^{VRP}$，工厂位置信息
输出: 单车辆运输解(抵达各工厂的时间)
$count\_n\leftarrow 0, L\leftarrow 0$
Repeat
  $count\_n\leftarrow count\_n+1$
  If $count\_n\ne n\_p+1$ **then**
    $i\leftarrow \pi_{count\_n}^{VRP}$
  Else
    $L\leftarrow L+$ 配件厂 i' 与装配厂之间距离
    $total\_T\leftarrow L/speed$
  End if
  If $count\_n=1$ **then**
    $L\leftarrow$ 配件厂 i 与装配厂之间距离
  Else
    $L\leftarrow L+$ 配件厂 i 与 i' 之间距离
  End if
  $T_i^{VRP}\leftarrow L/speed$，$i'\leftarrow i$
Until $count\_n=n\_p+1$

SALBP 和 VRP 编码方式决定这两类问题在解码时均不需要修正编码排序，编码排序即为装配排序和车辆抵达工厂的排序. SALBP 和 VRP 的解码方法具体如算法 6、算法 7. 评价装配解即计算装配解对应的装配节拍. 装配节拍由完工时间最大工作站完工时间决定. 评价运输解即计算车辆从出发至返回装配厂的总时间，其计算方法已包含于算法 7 VRP 的解码之中，$total\_T$ 表示单车辆运输总时间.

3.4.3 LVNS 求解两类子问题

在 2.1.2 节中，已分析了装配导向的分解策略的合理性. 该策略将原问题分解为简单装配线平衡问题(SALBP)和路径规划问题(VRP). 针对这两类具有相似结构的问题，采用本文设计的 LVNS 分别对其求解. 求解的一般过程如算法 8.

算法 8 LVNS 算法求解子问题的通用框架

输入: 参数设置，问题实例
输出: 优质解 X^*
初始化问题编码 Π'，$P_k(k=1,2,3,4)$，$Flag\leftarrow 0$
历史最优解 $X^*\leftarrow$ 解码 Π'
历史最优评价值 $value^*\leftarrow$ 评价 X^*
历史最优编码 $\Pi^*\leftarrow\Pi'$
Repeat
  If $Flag=n\_A$ **then**
    $NA_s\leftarrow$ 以最优策略选择动作，$Flag\leftarrow 0$
    $\Pi''\leftarrow$ 对 Π'' 执行 NA_s 动作
  Else
    $NA_s\leftarrow$ 执行 NA 选择函数
    $\Pi''\leftarrow$ 对 Π^* 执行 NA_s 动作
  End if
  $X''\leftarrow$ 解码 Π''，$value''\leftarrow$ 评价 X''
  If $value''<value^*$ **then**
    $X^*\leftarrow X'', value^*\leftarrow value'', \Pi^*\leftarrow X'', Flag\leftarrow 0$
  Else
    $Flag\leftarrow Flag+1$
  End if
  $P_k\leftarrow$ 执行 P_k 的更新函数
Until 程序运行时间 $>set\_t$

$$LB=\max\left(t_{\max},\frac{\sum_{i\in S^{alb}}t_i}{m}\right) \tag{36}$$

算法 8 中, 不同问题对计算时间的需求不同. 例如, SALBP 中可采用 $n\times m$ 作为计算时间最大值[23]. VRP 中可采用 $a\times n\_p$ 作为计算时间最大值, a 通常取 1~3[6], n_p 表示单车辆需途径的任务点数目. 因此, 本文 LVNS 算法求解 SALBP 和 VRP 时, set_t 分别取 $n\times m$ 和 $2\times n\_p$. 特别之处, 应用 LVNS 求解 SALBP 时, 解码 Π' 或 Π'' 需要预设节拍约束才能实现解码. 本文中, 解码 Π' 和 Π'' 时预设的节拍约束分别为 $2\times LB$ [23]和 $value^*-1$, 前者用于首次解码, 后者则使用压缩的已发现最优节拍来寻找满足该装配节拍的更优值. LB 表示 SALBP 装配节拍的下界, 可由公式(36)计算. $Flag$ 值表示连续 $Flag$ 代问题的优化目标没有改进. 本文设置当 $Flag$ 与邻域动作种类数相同时启用扰动策略.

3.5 LVNS_DS 求解 TACOP

算法 9 LVNS_DS 算法求解 TACOP

输入: 算例及参数设置
输出: 满足问题约束和优化目标关系的优质解
$CT=2\times LB$, $t\leftarrow 0$
Repeat
  LVNS 算法优化 SALBP
Until $t\geq n\times m$ 秒
由算法 1 对 Π^{ALB} 扩展得到 Set, $t\leftarrow 0$
Repeat
  $\Pi^{ALB'}\leftarrow$ 从 Set 中随机取装配解, $n^{car}\leftarrow n^{car}_{\min}$
  Repeat
    按算法 2 将 $\Pi^{ALB'}$ 分解为 n^{car} 个 VRP 问题
    For $k=1$ **to** n^{car} do
      Repeat
        LVNS 算法优化第 k 个 VRP
      Until $t\geq 2\times n\_p$ 秒
    End for
    计算 $TC,\overline{T_{re}}$,更新 TACOP 优质解, $n^{car}\leftarrow n^{car}+1$
  Until $n^{car}=n$
Until $t\geq n\times\left(n-n^{car}_{\min}+1\right)$ 秒

应用融合 ADS 分解策略的 LVNS 算法对 TACOP 进行整体求解. 首先, 利用 ADS 分解策略将 TACOP 分解为一个 SALBP 和多个 VRP. 其次, 采用 LVNS 求解 SALBP 后得到优质装配解. 通过算法 1 将优质装配解扩充为包含优质装配解的集合 Set. 然后从 Set 中任取装配解对该解应用算法 2 后获取 n^{car} 辆运输车的任务集 $S^{car}_k\left(k=\left(1,2,...,n^{car}\right)\right)$. 采用 LVNS 算法对 n^{car} 个 VRP 进行独立求解并得到每辆车的优质运输解 Π^{VRP}. 最后, 通过给定时间下的迭代搜索, 输出生产节拍 CT 为主要优化目标, 运输成本 TC 、平均配件滞留时间 $\overline{T_{re}}$ 非支配集为次要优化目标的优质解, 求解非支配解时设置程序最大运行时间与装配工序数目、车辆取值范围成正相关, 设置为 $n\times\left(n-n^{car}_{\min}+1\right)$. 具体过程如算法 9.

4 实验分析

实验部分主要包括算法关键环节验证、算法整体性能验证. 关键环节验证设置 2 组对照实验, 分别验证装配导向式分解策略有效性和运输-装配协同优化重要性. 算法整体性能有效性验证同样设置 2 组对照实验, 分别选取 VNS 算法及主流多目标算法 NSGA-2、MOEA/D 与 LVNS_DS 进行实验对比. 实验设备处理器为 AMD_Ryzen7_2700U, 主频为 2.20 GHz, RAM 为 8.00GB, 操作系统为 Win10. 编程语言均为 Python3.8. 在以上实验中, 实验结果取每种算法对每种算例运行 20 次后的平均值.

4.1 算例介绍

TACOP 算例主要由配件厂位置坐标信息、装配工序优先关系、配件重量、装配线拥有工作站数目以及车辆信息等构成. 目前, 国内外学术期刊上暂无求解 TACOP 的公开算例. 本文基于公开的装配线平衡问题算例(取自专业数据支持网 https://assembly-line-balancing.de), 对其补充配件厂位置坐标信息、车辆信息、配件重量后构成 TACOP 的实验算例. 具体方法为, 装配厂位置坐标设置为 $(0,0)$. 配件厂位置坐标 (x,y) 是以装配厂为中心随机生成, $(x,y)\in S_{factory}$, 单位为公里. 设装配厂存在

10 条并行装配线, 即 $nl=10$. 不同算例所需装配线的工作站数目如表 3 所示. 零件重量设置为 $\{m|m\in[5,10]\}$, 单位为千克. 表 4 为该算例车辆信息设置.

$$S_{factory}=\{(x,y)|x,y\in[-50,50],x,y\neq 0\} \quad (37)$$

表 3 不同产品装配线的工作站数目
Tab.3 Number of workstations for different product assembly lines

装配算例	m
JAESCHKE	4
JACKSON	5
BUXEY	6
KILBRID	8
LUTZ1	10
LUTZ2	40

表 4 车辆参数
Tab.4 Vehicle parameters

最大载重	每公里费用	固定费用	平均车速
800kg	2.5 元	600 元	45km/小时

4.2 算法评价指标

本文模型中首先将三个优化目标划分为主次目标, 即装配节拍为主要目标, 运输成本、库存水平为的非支配解为次要目标. 因此, 本文中同一算法不同解之间、不同算法解之间的“支配关系”是按上述关系进行对比, 即在评价算法性能时, 首先需要比较不同解中主要目标的大小, 然后比较次要目标的支配关系, 也只有在主要目标相同时, 才比较次要目标.

不同算法求解同一问题得到的非支配解集可以在二维或三维空间中定性地对比, 也可使用更加严谨的量化指标对算法性能做对比. 本文采用 Ishibuchi 等[24][25]提出的评价方法对算法定量评估. 该评价指标包括 R_N 和 N_N 两部分, R_N 表示算法获取非支配解的占优比, N_N 表示算法获取非支配解的占优数. 两个指标的计算过程如下:

(1)针对同一问题, 将 k 种不同算法获取的非支配解集 $S_1,S_2,...,S_j,...,S_k$ 合并为一个集合 S_{sum}.

(2)以主次目标顺序计算 j 算法获取结果的占优个数 $N\_N(S_j)$, 即统计 S_j 内元素不受 S_{sum} 内元素支配的元素数量, 如公式(38)所示.

(3)计算 j 算法的占优比 $R\_N(S_j)$, 即利用 S_j 中不受 S_{sum} 支配的元素数目除以 S_{sum} 元素总数目得到, 如公式(39)所示.

$$N\_N(S_j)=\left|S_j-\{x\in S_j\mid\exists y\in S_{sum},x\ \mathrm{p}\ y\}\right| \quad (38)$$

$$R\_N(S_j)=\frac{N\_N(S_j)}{|S_{sum}|} \quad (39)$$

4.3 参数实验

LVNS_DS 算法关键参数包括邻域动作更新函数中奖励系数 α 和惩罚系数 β. 本文采用实验设计(Design of Experiments, DOE)法对两参数进行设置. 参数水平设置如表 5. 取 R_N 作为各参数水平的响应值进行比较. 具体响应值如表 6、表 7.

表 5 参数水平设置
Tab.5 Parameter level setting

参数	水平		
α	0.1	0.3	0.5
β	0.2	0.4	0.6

表 6 参数设置及响应值
Tab.6 Parameter setting and response value

组别	α	β	响应值
1	1	1	0.013
2	1	2	0.015
3	1	3	0.026
4	2	1	0.023
5	2	2	0.014
6	2	3	0.014
7	3	1	0.030
8	3	2	0.020
9	3	3	0.019

表 7 参数水平及响应值
Tab.7 Parameter level and response value

水平	α	β
1	0.018	0.022
2	0.017	0.016
3	0.023	0.020
极差	0.006	0.005
影响等级	1	2

其中，表 6 为各参数在不同取值组合下的实验结果，表 7 是在表 6 基础上，进一步计算了各参数在不同参数水平(即取值)下 R_N 的平均值，例如，参数 α 取 1 水平(即 0.1)时，由表 6 可计算参数 α 在 1 水平下 R_N 的平均值为 0.018. 最终通过表 7 中数据确定各参数的取值.

参数实验结果表明，存在奖励和惩罚系数的极小值使算法性能达到较优状态. 奖励与惩罚系数的变化趋势对算法的影响趋势基本一致，但惩罚系数的影响略大. 原因在于，惩罚系数可以防止算法对单一尺度邻域动做持续选择，促进算法跳出局部最优；而奖励系数对算法起引导作用，可引导算法大概率地选择最佳邻域动作，使算法在不同搜索阶段应用不同尺度邻域动作对解空间进行搜索，从而提升算法搜索效率. 在参数实验结果分析基础上，取 $\alpha=0.3$，$\beta=0.4$ 作为VNS_DS算法的参数取值.

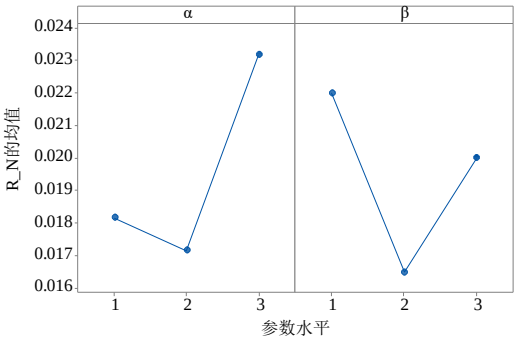

图 6 参数设置响应曲线
Fig.6 Parameter setting response curve

4.4 算法关键环节验证

关键环节实验包括验证装配导向式分解策略(ADS)有效性和运输-装配协同优化有效性. 为此，分别设计包含运输导向式分解策略(TDS)的变形算法 LVNS_TO 和优质装配解确定的变形算法 LVNS_GS，将变形算法与 LVNS_DS 算法进行实验对比.

4.4.1 验证装配导向式分解策略有效性

表 8 是包含两种不同分解策略的实验对比结果，表中 NB 表示某指标下算法在所有算例中取得最好结果的个数. 比较两种分解策略，ADS 在 R_N 和 N_N 两种评价指标下均取得全部最好值. 其原因在于，ADS 首先能够保证装配环节达到一种较优状态；其次，由于优质装配解存在不唯一性(见性质 1)，所以在 ADS 下运输-装配环节不是独立优化，而是一种协同优化. 此外，LVNS_DS 算法能够在已知装配解和运输解后，借助配件在途时长和工序开始装配时间逆向地计算车辆出发时间. 因此，算法能够通过决策车辆出发时间来减少车辆过早或过晚地将配件运抵装配厂的现象，使多数配件能够较为准时地抵达装配线，从而降低企业库存成本，在计算指标上则表现为配件平均滞留时间减小. 而 TDS 仅能保证运输解较优，但无法决策车辆出发时间，难以很好地衔接配件抵达装配厂时间和配件需装装配的时间，使装配线易于产生配件短缺或配件滞留等问题，从而增加企业运营中的浪费现象，在计算指标上则表现为装配线节拍值过大或配件平均滞留时间较大，如图 7. 在实际生产中，装配线节拍值过大、运输成本较低的方案企业通常也不会采用，这是因为装配线节拍直接影响企业的生产效率，低效的装配线不利于后续生产或销售环节. 因此，ADS 较 TDS 更适于 TACOP.

4.4.2 验证运输-装配协同优化有效性

现有研究通常将装配解固定，然后研究如何优化装配线站边库存和供料过程的运输成本. 为验证运输-装配协同优化有效性，首先，提出并证明了优质装配解存在的不唯一性，然后，基于该性质设计融合装配导向式分解策略(ADS)的 LVNS_DS 算法对 TACOP 进行求解. 因此，验证运输-装配协同优化有效性即验证性质 1 对 TACOP 影响程度. LVNS_DS 算法求解 TACOP 时，运输与装配之间是通过一组包含优质装配解的集合建立了协同关系，所以其对比算法(LVNS_GS)设计为，运输与装配之间基于确定的优质装配解. 在对比算法中，除装配解是确定解以外，算法其余部分与本文 LVNS_DS 算法均相同. 实验结果如表 8.

表 8 LVNS_TO、LVNS _GS 与 LVNS_DS 的对比结果
Tab 8. Comparison results of LVNS_TO、LVNS _GS and LVNS_DS

算例	LVNS_TO		LVNS_GS		LVNS_DS	
	N_N	R_N	N_N	R_N	N_N	R_N
JACKSON	2.36	0.13	0.36	0.02	**7.64**	**0.40**
JAESCHKE	0.00	0.00	1.36	0.04	**2.09**	**0.15**
BUXEY	1.00	0.06	1.00	0.04	**9.91**	**0.48**
KILBRID	0.00	0.00	**13.91**	**0.37**	10.00	0.25
LUTZ 1	0.50	0.02	2.20	0.09	**17.70**	**0.59**
LUTZ2	0.45	0.02	11.09	0.21	**28.27**	**0.54**
NB	0	0	1	1	**5**	**5**

表 9 LVNS_DS 与 VNS、NSGA-2、MOEA/D 的对比结果
Tab.9 Comparison results of LVNS_DS、VNS、NSGA-2 and MOEA/D

算例	VNS		NSGA-2		MOEA/D		LVNS_DS	
	N_N	R_N	N_N	R_N	N_N	R_N	N_N	R_N
BUXEY	1.70	0.06	1.00	0.04	1.75	0.07	**2.25**	**0.08**
JACKSON	**5.20**	**0.11**	1.65	0.03	2.25	0.05	4.75	0.09
JAESCHKE	6.30	0.14	5.40	0.10	2.80	0.05	**6.75**	**0.13**
KILBRID	6.65	0.12	2.10	0.03	3.05	0.04	**7.10**	**0.11**
LUTZ1	14.75	0.13	6.00	0.05	10.10	0.08	**15.60**	**0.13**
LUTZ2	1.70	0.06	1.00	0.04	1.75	0.07	**2.25**	**0.08**
NB	1	1	0	0	0	0	**5**	**5**

由表 8, 考虑运输-装配协同优化策略的 LVNS_DS 较 LVNS_GS 明显占优. 原因在于, 优质装配解的不唯一性扩大了运输成本、库存水平的非支配解空间, 所以 LVNS_DS 能够取得更多非支配解. 相比之下, 单一且确定的优质装配解限制了运输成本、库存水平的解空间. 因此, 运输-装配协同优化不仅能够保证装配效率, 而且能够保证一定的运输、库存优化效果.

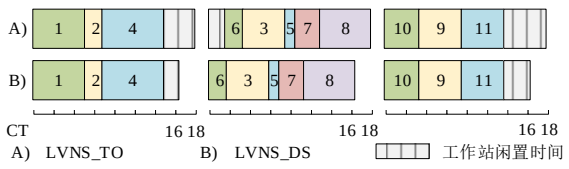

图 7 两种分解策略下装配解的计算结果示意图(P9)
Fig.7 Schematic diagram of calculation results of assembly solution under two decomposition strategies(P9)

4.5 与其它算法对比

为验证具有邻域学习能力的 LVNS_DS 算法在求解 TACOP 时具有较好的性能, 将其与主流多目标算法 NSGA-2 和 MOEA/D 以及标准 VNS 算法做对比. 为保证公平性, NSGA-2、MOEA/D 同样采用装配导向式分解策略. 标准 VNS 算法应用于该问题的实现过程与 LVNS_DS 算法相似, 区别仅在于邻域动作的选择方式.

本节实验的对比结果如表 9 所示. 在该结果中, LVNS_DS 算法表现最为出色, VNS 算法略逊于 LVNS_DS 算法, 而 NSGA-2 在求解 TACOP 时效果不佳. 其关键原因在于, 基于 GA 的 NSGA-2 在处理局部搜索时是应用交叉、变异算子, 这两种算子在求解 SALBP 与 VRP 子问题时的作用效果仅相当

于 VNS 算法中两种邻域操作, 同时, 由于 TACOP 解空间较 VRP、SALBP 更为复杂, 因此, 该算法不足以对 TACOP 解空间进行细致搜索. MOEA/D 存在相似的问题, 但 MOEA/D 较 NSGA-2 拥有更小的计算复杂度, 因此实验结果中 MOEA/D 比 NAGA-2 略好. 相比采用交叉变异的局部搜素方式, 基于单个体的 LVNS_DS 算法可根据问题性质, 采用不同尺度的邻域动作在解空间中迅速转换, 从而提升算法搜索效率. 此外, LVNS_DS 算法改进了 VNS 算法中邻域结构的转换方式, 使其通过学习函数和选择函数来降低邻域动作选择的盲目性, 使算法能够自适应地选取合适的邻域动作来执行不同阶段的搜索任务. 因此, LVNS_DS 算法相比 NSGA-2[26]、MOEA/D[27]和 VNS 算法在求解 TACOP 时拥有更好的计算性能.

5 结论

本文针对运输-装配协同优化问题(TACOP), 建立以 TC、$\overline{T_{re}}$ 和 CT 为优化目标的整数规划模型. 通过分析 TACOP 阶段间与阶段内部耦合性, 提出求解此问题的合理分解策略, 并设计一种融合该策略的学习型变邻域搜索算法(LVNS_DS)对问题进行求解. 在理论分析的基础上, 通过数值实验验证了 LVNS_DS 算法求解 TACOP 的有效性. 后续将进一步研究装配阶段为 U 型、双边等更为复杂、实际的装配线与运输、库存协同的优化问题, 以及在路径规划问题中考虑装载顺序约束与装配厂配件装配顺序的协同优化问题.

参考文献

[1] 徐颖, 刘勤明, 周林森. 基于博弈论的闭环双渠道回收供应链决策研究[J]. 系统仿真学报, 2022, 34(2):396-408.
Xu Ying, Liu Qinming, Zhou Linsen. Research on Decision-Making of Closed-Loop Supply Chain for Dual-Channel Recovery Based on Game Theory[J]. Journal of System Simulation, 2022, 34(2):396-408.

[2] 王玉. "物流+制造"共探新时代融合发展路径[J]. 物流技术与应用, 2021, 26(5):100-103.
Wang Yu. Exploring the Integrated Development Path in the New Era With "Logistics and Manufacturing"[J]. Logistics & Material Handling, 2021, 26(5):100-103.

[3] 张明超, 孙新波, 王永霞. 数据赋能驱动精益生产创新内在机理的案例研究[J]. 南开管理评论, 2021, 24(3):102-116.
Zhang Mingchao, Sun Xinbo, Wang Yongxia. A Case Study on the Internal Mechanism of Lean Production Innovation Driven by Data Empowerment[J]. Nankai Business Review, 2021, 24(3):102-116.

[4] Zhou Binghai, He Zhaoxu. A Static Semi-kitting Strategy System of JIT Material Distribution Scheduling for Mixed-flow Assembly Lines[J]. Expert Systems with Applications(S0957-4174), 2021, 184(4):115523

[5] Emde S, Boysen N. Optimally Routing and Scheduling Tow Trains for JIT-supply of Mixed-model Assembly Lines[J]. European Journal of Operational Research(S0377-2217), 2012, 217(2):287-299.

[6] 胡蓉, 陈文博, 钱斌, 等. 学习型蚁群算法求解绿色多车场车辆路径问题[J]. 系统仿真学报, 2021, 33(9):2095-2108.
Hu Rong, Chen Wenbo, Qian Bin, et al. Learning Ant Colony Algorithm for Green Multi-depot Vehicle Routing Problem[J]. Journal of System Simulation, 2021, 33(9):2095-2108.

[7] Miranda D M, Concei O S V. The Vehicle Routing Problem with Hard Time Windows and Stochastic Travel and Service Time[J]. Expert Systems with Applications(S0957-4174), 2016, 64:104-116.

[8] Zhang D, Cai S, Ye F, et al. A Hybrid Algorithm for a Vehicle Routing Problem with Realistic Constraints[J]. Information Sciences(S0020-0255), 2017, 394-395:167-182.

[9] Koulamas Christos, Kyparisis George J. The No-wait Flow Shop with Rejection[J]. International Journal of Production Research(S0020-7543), 2021, 59(6):1852-1859.

[10] Driscoll J, Thilakawardana D. The Definition of Assembly Line Balancing Difficulty and Evaluation of Balance Solution Quality[J]. Robotics and Computer Integrated Manufacturing(S0736-5845), 2001, 17(1-2):81-86.

[11] Scholl B A. A Survey on Problems and Methods in Generalized Assembly Line Balancing[J]. European Journal of Operational Research(S0377-2217), 2006, 168(3):694-715.

[12] 邓超, 钱斌, 胡蓉, 等. 融合规则的 HEDA 求解带工件批量运输的三阶段装配协同优化问题[J]. 控制与决策, 2020, 35(10): 2507-2513.
Deng Chao, Qin Bin, Hu Rong, et al. Rule-based Hybrid EDA for Three-stage Assembly Integrated Scheduling

Problem with Job Batches Transportation[J]. Control and Decision, 2020, 35(10):2507-2513.

[13] 周炳海, 彭涛. 混流装配生产线准时化物料补给调度方法[J]. 控制与决策, 2017, 32(06):976-982.
Zhou Binghang, Peng Tao. Scheduling Methods of Just-in-time Material Replenishment in Mixed-model Assembly Lines[J]. Control and Decision. 2017, 32(06):976-982.

[14] Fathi M, Rodriguez V, Fontes D B M M, et al. A Modified Particle Swarm Optimisation Algorithm to Solve the Part Feeding Problem at Assembly Lines[J]. International Journal of Production Research(S0020-7543), 2017, 54(3-4):1-16.

[15] Golz J, Gujjula R, Ho Günther, et al. Part Feeding at High-variant Mixed-model Assembly Lines[J]. Flexible Services and Manufacturing Journal(S1936-6582), 2010, 24(2):119-141.

[16] Rahman H F, Janardhanan M N, Nielsen P. An Integrated Approach for Line Balancing and AGV Scheduling towards Smart Assembly Systems[J]. Assembly Automation(S0144-5154), 2020, 40(2):219-234.

[17] Mladenovi N, Hansen P. Variable Neighborhood Search[J]. Computers & Operations Research(S0305-0548), 1997, 24(11):1097-1100.

[18] 严洪森, 万晓琴, 熊福力. 基于 VNS-EM 混合算法的两阶段装配流水车间调度[J]. 东南大学学报(自然科学版), 2014(44):1285-1289.
Yan Hongsen, Wan Xiaoqin, Xiong Fuli. Two-stage Assembly Flow Shop Scheduling Based on Hybrid VNS-EM Algorithm[J]. Journal of Southeast University (Natural Science Edition), 2014, 44(6):1285-1289.

[19] 周炳海, 费芊然. 考虑能耗的混流装配线物料配送多目标调度方法[J]. 东北大学学报:自然科学版, 2020, 41(2): 258-264.
Zhou Binghai, Fei Qianran. Multi-objective Scheduling Algorithm for Mixed-Model Assembly Line Considering Energy Consumption[J]. Journal of Northeastern University Natural Science, 2020, 41(2): 258-264.

[20] 南丽君, 陈彦如, 张宗成. 改进的自适应大规模邻域搜索算法求解动态需求的混合车辆路径问题[J]. 计算机应用研究, 2021, 38(10):2926-2934.
Nan Lijun, Chen Yanru, Zhang Zongcheng. Improved Adaptive Large Neighborhood Search Algorithm for Mixed Fleet Routing Problem of Dynamic Demands[J]. Application Reasearch of Computers, 2021, 38(10):2926-2934.

[21] Hansen P, Mladenovi N. Variable Neighborhood Search: Principles and Applications[J]. European Journal of Operational Research(S0377-2217), 2001, 130(3):449-467.

[22] Prajapati A, Chhabra J K. Information-Theoretic Remodularization of Object-Oriented Software Systems[J]. Information Systems Frontiers(S1387-3326), 2020, 22(4):863-880.

[23] Tang Q, Li Z, Zhang L. An Effective Discrete Artificial Bee Colony Algorithm with Idle Time Reduction Techniques for Two-sided Assembly Line Balancing Problem of Type-II[J]. Computers & Industrial Engineering(S0360-8352), 2016, 97(3):146-156.

[24] Ishibuchi H, Yoshida T, Murata T. Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling[J]. IEEE Transactions on Evolutionary Computation(S1089-778X), 2003, 7(2):204-223.

[25] 郑逸凡, 钱斌, 胡蓉,等. CE-GA 协同进化算法求解人机共同作业的 U 形装配线平衡问题[J]. 机械工程学报, 2020, 56 (9):199-214.
Zheng Yifan, Qian Bin, Hu Rong, et al. CE-GA Co-evolutionary Algorithm for Solving U-shaped Assembly Line Balancing Problem with Man-robot Cooperation[J]. Journal of Mechanical Engineering, 2020, 56(9):199-214.

[26] Deb K, Pratap A, Agarwal S, et al. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II[J]. IEEE Transactions on Evolutionary Computation(S1089-778X), 2002, 6(2):182-197.

[27] Zhang Q, Hui L. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition[J]. IEEE Transactions on Evolutionary Computation(S1089-778X), 2008, 11(6):712-731.