计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(10):183−189 [doi: 10.15888/j.cnki.csa.007096]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于突破性局部搜索的集装箱班列同步转运作业调度优化①

何　迅, 郭　鹏, 栾玉麟

(西南交通大学 机械工程学院, 成都 610031)
(轨道交通运维技术与装备四川省重点实验室, 成都 610031)
通讯作者: 郭　鹏, E-mail: pengguo318@swjtu.edu.cn

摘　要: 铁路集装箱中心站作为内陆腹地运输网络中的重要节点. 转运作为主要的站内作业活动, 对其开展调度研究, 能够有效缩短不同运输方式之间的转运周期, 从而保证铁路集装箱多式联运的整体运作效率. 为了快速制定堆场作业计划, 现有铁路集装箱站场转运作业均要求班列间的集装箱转运须经堆场方可实现. 在该模式下, 必然发生两次装卸和一次暂存作业. 为了避免暂存中转作业, 采用集装箱班列同步转运作业模式构建了以同步转运集装箱箱数最大化为目标的数学规划模型, 提出突破性局部搜索框架实现对大规模问题的求解. 最后通过算例分析验证了所提出的启发式算法效率.

关键词: 同步转运; 铁路集装箱中心站; 调度优化; 突破性局部搜索

引用格式: 何迅,郭鹏,栾玉麟.基于突破性局部搜索的集装箱班列同步转运作业调度优化.计算机系统应用,2019,28(10):183–189. http://www.c-s-a.org.cn/1003-3254/7096.html

Breakout Local Search for Scheduling Freight Trains with Synchronized Transferring Operations

HE Xun, GUO Peng, LUAN Yu-Lin

(School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China)
(Technology and Equipment of Rail Transit Operation and Maintenance Key Laboratory of Sichuan Province, Chengdu 610031, China)

Abstract: Railway container terminals are regarded as the important nodes in hinterland transportation network. Transferring operation is main activity in operations management of container terminals. Scheduling the transferring activities optimally can be able to shorten the operating cycle among different transport vehicles, and improve the efficiency of cargo transportation in rail multimodal transportation. To make the operation plan of stacking yard rapidly, the transferring operation requires the container movement must be stacked intermediately via yard at present. In that scenario, two loading/discharging activities and one stacking activity must be occurred. For avoiding idly extra activities, synchronized transferring should be recommended. In this study, a mixed integer programming model with maximizing the number of synchronized transferring containers is proposed. Due to its intractability, a breakout local search algorithm is designed to solve this problem under study, especially for large-sized instances. The computational experiments are performed to evaluate the performance of the proposed algorithm.

Key words: synchronized transferring; railway container terminal; scheduling; breakout local search

① 基金项目: 国家自然科学基金 (51405403); 中央高校基本科研业务费专项资金 (2682018CX09)
Foundation item: National Natural Science Foundation of China (51405403); Fundamental Research Funds for the Central Universities (2682018CX09)
收稿时间: 2019-03-19; 修改时间: 2019-04-17; 采用时间: 2019-04-19; csa 在线出版时间: 2019-10-15

铁路运输作为最为低碳的货运方式, 在内陆腹地运输方面极具优势. 我国铁路货运总量占比在面临残酷的市场竞争时连年下滑, 已由 2009 年的 11.96% 降至 2017 年的 7.81%[1]. 目前我国铁路集装箱运量仅占铁路货运量的 5.4%, 远远低于发达国家 (美国占比 49%, 法国占比 40%, 英国占比 30%, 德国占比 20%, 日本占比 100%)[2]. 如此大的差距表明我国铁路集装箱运输存在巨大的潜力和发展空间. 当前, 我国沿海港口积极发展内陆“无水港”, 加快港口铁路集疏运, 积极推动船、车、班列、港口、场站、货物等信息开放共享. 国家“十一五规划”规划建设了 18 个铁路集装箱中心站, 有利于实现区域集装箱铁路运输与其他运输方式的无缝衔接. 在“一带一路”国家战略的带动下, 铁路集装箱多式联运更是迎来了弥足珍贵的发展契机, 研究提升铁路集装箱多式联运效率具有重要的现实与战略意义.

作为多式联运铁路站场主要活动, 转运作业效率直接影响铁路集装箱多式联运的整体运作水平. 为了保证自动化堆场作业计划生成的准确性, 现有铁路集装箱站场转运作业均要求集装箱班列或卡车经堆场方可实现. 经集装箱站场实现不同运载工具之间的转运必然存在两次装卸和一次暂存作业. 对于提高转运作业效率来说, 暂存和额外的装卸作业完全是无效操作, 且会使有限的堆场空间资源更为紧张. 减少暂存中转作业的关键是提高同步转运 (也叫连续转运或直装直卸) 集装箱比率. 同步转运旨在通过合理安排集装箱班列进入堆场的作业时间窗, 实现在站的两类运输设备之间的集装箱流同步交互, 尽可能地缩短集装箱站内中转时间.

国内外学者对铁路集装箱站场的作业效率提升进行了不少研究. Boysen 等[3]对铁路站场的设施布局与作业调度优化研究现状进行了归类总结. Otto 和 Pesch[4]分析了多式联运下铁路堆场中班列与堆场指派问题, 以最小化再指派次数为目标更新了下界计算方法. Boysen 等[5]构建了铁路-铁路 (Rail-Rail) 站场的数学规划模型来确定主装卸线上各台起重机的最优作业区域. Kellner 等[6]考虑集装箱班列停靠位置对转运作业的影响, 并构建数学模型来确定其最佳位置以便减少集装箱移动距离. Boysen 等[7,8]提出采用集装箱整理系统支持站内转运, 并对比分析了四种不同的集装箱整理系统在铁路-铁路联运站场中的作业效率. 起重机作为集装箱站场的主要转运设备, Guo 等[9,10]对公铁联运的集装箱中心站轨道式起重机调度问题进行了建模优化, 并希望将外部卡车服务水平纳入调度优化模型. 王力等[11,12]认为集装箱班列卸箱作业和装箱作业过程相同, 以卸箱作业过程建立了中心站起重机调度优化模型, 设计了混合粒子群算法进行求解; 还考虑堆场混堆优化问题, 以两阶段优化模型来平衡堆场各箱区进站箱和出站箱的数量及减少堆存所产生的压箱数. 而上述研究聚焦在铁路站场的箱位指派、集装箱班列位置指派、装卸设备调度问题上, 尚未考虑同步转运作业.

在不同的运输方式之间利用转运工具实现乘客或货物中转, 尽可能保证进出运载工具的同步, 能够大幅提升转运效率[13]. Boysen 等[14]为铁路-铁路 (Rail-Rail) 集装箱站场调度问题构建了数学模型, 通过确定集装箱班列集合中每趟车的服务间隙来最小化班列重回站场的次数和集装箱分割装卸的移动次数, 证明了该问题是 NP-hard 的, 并提出了基于动态规划和束搜索的求解算法. 该研究对集装箱班列的转运调度进行优化, 以求实现同步转运, 本文在此基础上提出集装箱班列同步转运调度模型, 通过分析问题的特征设计基于突破性局部搜索的求解算法. 利用算例验证本文算法可以在提高求解质量的同时, 显著缩短求解时间, 对站场效率提升具有重要的参考价值.

1 问题描述与数学模型

在集装箱班列同步转运作业调度问题 (Freight Trains Scheduling Problem with Synchronized Transferring, FTSPST) 中, n 趟班列需要进入堆场实施装卸转运操作. 堆场铺设有 m 组并行铁轨, 每次最多可牵引 m 趟班列进入. 假设有 T 个时间段实施牵引进站作业, 则有 $T=n/m$. 为了方便描述, 在此假设 $n=T\times m$. 在实际作业过程中, 若班列数不足 n, 可通过引入未装载集装箱的虚拟班列来满足. 针对班列 i (i=1, …, n), 最早进入堆场时间段为 e_i, 最晚离开堆场时间段为 l_i. 假设班列 i 和 j 之间存在 A_{ij} 个集装箱须进行转运, 若两趟班列能够在同一时间段 t 牵引进堆场, 则可成功实现 A_{ij} 个集装箱同步转运. 需要转运到班列 i 的集装箱位于其他班列之上, 用 L_i 表示能与班列 i 进行同步转运作业的其他班列集合. 在此定义决策变量 x_{it} 与 z_{ij}, 若集装箱班列 i 进入堆场的时间段为 t 则 x_{it}=1, 否则为 0; 若集装箱班列 i 和 j 在同一时间段进入堆场则 z_{ij}=1, 否则为 0. 以同步转运的集装箱箱数最大化为优化目标,

构建如下混合整数规划模型:

$$\text{Max} f = \sum_{i=1}^{n}\sum_{j=1}^{n} z_{ij}A_{ij} \tag{1}$$

s.t.

$$\sum_{t=e_i}^{l_i} x_{it} = 1, \forall i = 1,\cdots,n \tag{2}$$

$$\sum_{i=1}^{n} x_{it} \le m, \forall t = 1,\cdots,T \tag{3}$$

$$\left|\sum_{t=1}^{T} x_{it}\times t - \sum_{t=1}^{T} x_{jt}\times t\right| \le (1-z_{ij})\cdot M, \forall i,j = 1,\cdots,n \tag{4}$$

$$x_{it}, z_{ij} \in \{0,1\}, \forall i,j = 1,\cdots,n, t = 1,\cdots,T \tag{5}$$

上述优化模型中, 目标函数 (1) 保证同步转运的集装箱箱数最大化; 约束 (2) 确保一辆班列只能被安排在一个时间段; 约束 (3) 确保同一时间段内进入堆场的班列数不超过铺设的轨道数; 约束 (4) 表示同一时间段进入堆场的班列 i 和 j 对应的变量 z_{ij} 必须为 1; 约束 (5) 定义决策变量的取值范围.

2 突破性局部搜索算法

突破性局部搜索算法 (Breakout Local Search, BLS) 是一种自适应的邻域搜索算法, 在一般邻域搜索算法的基础上增加了自适应扰动机制, 能快速逃离局部最优以及提高稳定性[15]. 该算法分为局部搜索阶段和扰动阶段. 通过某种方法产生初始可行解 π_0, 利用邻域搜索算子产生所有的可行解, 从中找出局部最优解 π. 然后通过适当的扰动对当前解 π 进行操作以帮助算法逃离当前的局部最优解, 扰动获得的解将成为下一轮邻域搜索的起点. 上述搜索一直迭代直到达到算法终止条件为止. BLS 算法在搜索过程中融入了局部迭代搜索、禁忌搜索与模拟退火算法的操作机制, 具有搜索质量高、计算时间短等优点, 现已用于求解最大团问题[16]、最大割问题[17]、二次指派问题[15]、装配序列规划问题[18]以及门分派问题[19]等, 并获得较好的求解效果. BLS 算法流程图如图 1 所示.

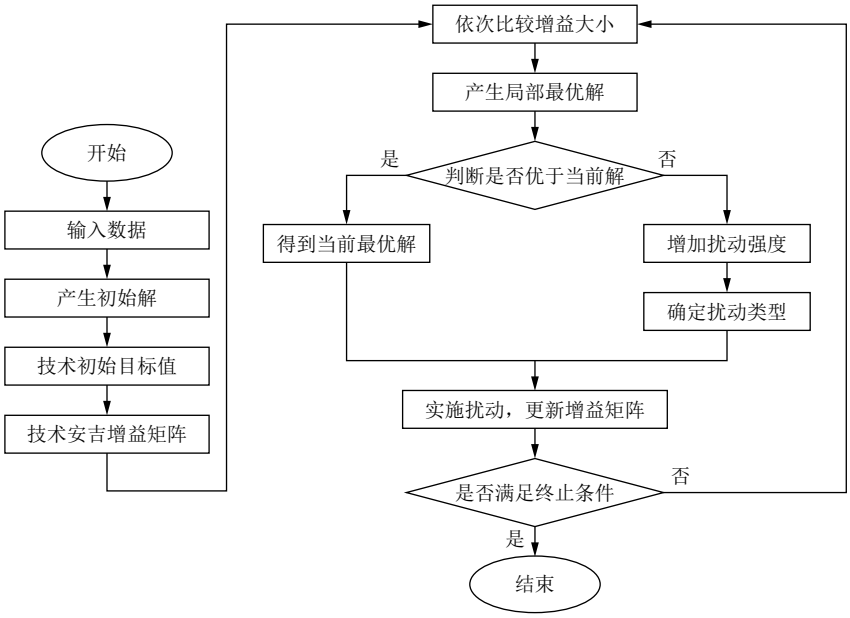

图 1 BLS 算法流程图

2.1 自适应扰动机制

BLS 算法对局部最优解的扰动取决于扰动幅度和扰动类型, 扰动的强弱依赖于当前搜索操作的状态, 主要是通过当前解 π 和当前最佳解连续未改善的次数 ω 来确定扰动幅度 K 和扰动类型[16]. BLS 首先搜索邻近的局部最优解, 只有当搜索停滞不前, 才能移动到下一个邻域. 在每次局部搜索后, BLS 执行小幅度的定向或近似随机移动, 以保证能够逃离当前局部最优解. 如果扰动幅度不足以逃离局部最优解, 则增加扰动幅度 K; 否则将其设置为默认值, 即 K=K_0. 如果当前最佳解

连续未改善的次数超过了给定的阀值 Ω, 则将扰动幅度设置为最大值 K_{max}, 以保证搜索过程能够脱离当前邻域解空间.

所提出的 BLS 算法采用置换序列的编码形式, 譬如 5 趟班列与 2 组铁轨的算例, 则当前解序列 π={1, 5, 2, 3, 4}, 其中 π_2=5 表示编号为 5 的班列安排在序列位置 2 处, 若班列 1 与班列 5 均可在时间段 t=1 进入堆场, 则其进入堆场的时间段为 1. 邻域操作与扰动采用交换操作, 则 $\pi'=\pi\oplus \text{swap}(u, v)$ 表示交换位置 u 和 v 上的班列序号得到新的候选解.

BLS 算法采用以下三种扰动类型: 定向, 近似随机和随机扰动. 定向扰动基于禁忌搜索原则来实施, 当搜索到的候选解优于目前为止找到的最佳解时, 该操作对应的禁忌状态消除. 同时产生候选解的操作在禁忌周期 γ 之外时, 也可消除禁忌状态. 定向扰动依赖于上一次迭代搜索的历史信息以及不会禁止良好的扰动操作. 针对定向扰动的判断由集合$\mathbb{R}$表征:

$$\mathbb{R}=\{\text{swap}(u,v)|\max\{\delta(\pi,u,v)\},(H_{uv}+\gamma)<Iter \text{ 或} (\delta(\pi,u,v)+f)<f_{\text{best}},u\neq v且1\le u,v\le n\} \quad (6)$$

其中, H 是上次执行扰动时跟踪迭代次数的矩阵, $Iter$ 是当前迭代次数, f 是当前解对应的目标函数值, f_{best} 是当前已找到的最佳解对应的目标函数值, u、v 为需要进行置换扰动的变量. γ 值越大代表着扰动越强烈, 在 BLS 算法中 $\gamma=n\times r_1+rand(0, 1)\times n\times r_2$, 其中 r_1 与 r_2 为待确定的常数值. 近似随机扰动仅依赖于由 H 矩阵提供的历史信息, 由集合$\mathbb{Z}$表征为:

$$\mathbb{Z}=\{\text{swap}(u,v)\left|\max\{H_{uv}\},u\neq v且1\le u,v\le n\right.\} \quad (7)$$

随机扰动简单地执行随机均匀的扰动, 由集合$\mathbb{C}$表征:

$$\mathbb{C}=\{\text{swap}\{u,v\}\left|u\neq v且1\le u,v\le n\right.\} \quad (8)$$

为了实现强化搜索和多样性搜索的平衡, BLS 算法引入概率选择操作确定扰动类型. 根据当前搜索状态和当前最佳解连续未改善的次数 ω 动态确定应用特定扰动的概率. 在迭代搜索过程中, 若找到新的最佳解或 ω 超过给定阈值 Ω 时, ω 被重置为零. 邻域搜索初期算法更偏向于采用定向扰动. 随着 ω 的增加, 使用定向扰动的几率逐渐降低, 同时随机和近似随机扰动的概率会增加, 从而达到自适应的多样性扰动.

另外, 从计算分析中已经观察到, 保证定向扰动的最小概率是有用的. 因此, 采用定向扰动的概率 p 取不小于阈值 p_0 的值:

$$p=\begin{cases} e^{-\frac{\omega}{\Omega}}, & \text{如果} e^{-\frac{\omega}{\Omega}}>p_0 \\ p_0, & \text{其他情况} \end{cases} \quad (9)$$

基于计算出的定向扰动概率 p, 则可分别通过 $(1–p)*Q$ 和 $(1–p)*(1–Q)$ 确定近似随机和随机扰动的概率, 其中 Q 为区间[0, 1]之间的常数. 确定了各自的概率后, 则可通过区间位于 (0, 1) 之间随机数来确定扰动类型.

2.2 基于问题的算法改进

铁路集装箱班列同步转运过程中, 存在每辆班列最早与最晚离开堆场的时间段和现有铁轨数的约束. 为此需对 BLS 算法进行必要的改进以适应所考虑的问题. 为了加快算法搜索速度, 在此设计算法 1 中的 BLS 初始可行解产生算法.

算法 1. BLS 初始解生成流程

输入: 班列间同步转运的集装箱箱数矩阵 A, 每辆班列的最早进入或最晚离开堆场时间段, 待安排的班列集合 Θ.
初始化: 将决策变量 x 和 z 置为 0 矩阵.
1. 根据班列最晚离开堆场的时间段对班列进行升序排列.
2. 班列进入堆场的时间段分为 T 个.
3. 时间段计数器 t=1.
4. while 待安排的班列集合 Θ 不为空 do
5. 从集合 Θ 中依次选择 m 趟班列.
6. if 选定的 m 趟班列能在时间段 t 进入堆场
7. 让 m 辆班列依次进入堆场.
8. 从集合 Θ 删除已进入堆场的班列序号.
9. end if
10. 更新 m 趟班列间能实施集装箱同步转运的 0-1 矩阵 z 和时间段指派矩阵 x.
11. t=t+1.
12. 计算成功实施同步转运的集装箱箱数.
13. end while
14. 输出: x 和 z.

邻域搜索与扰动产生的候选解对应的目标函数值计算将耗费大量的计算成本. 为了缩短计算时间, 在此依据每次班列置换后的增益来计算候选解的目标函数值. 针对数学规划模型的特点确定班列置换判断条件. 当班列 i 与班列 j 进行置换时, 班列 i 与 j 必须满足置换之前各自被分配在不同的时间段, 并且置换后分配的时间段不能超出各自最早进入和最晚离开堆场时间段. π'是当前解 π 通过交换位置 r 和 s 上的班列所得到的邻域解. 则当前解 π 与邻域解 π'之间的增益用式 (10) 计算:

$$\delta(\pi,r,s)=A_{rr}(z_{\pi_s\pi_s}-z_{\pi_r\pi_r})+A_{rs}(z_{\pi_s\pi_r}-z_{\pi_r\pi_s})+A_{sr}(z_{\pi_r\pi_s}-z_{\pi_s\pi_r})+A_{ss}(z_{\pi_r\pi_r}-z_{\pi_s\pi_s})+\sum_{i=1,i\neq r,s}^{n}\left(A_{ir}(z_{\pi_i\pi_s}-z_{\pi_i\pi_r})+A_{is}\left(z_{\pi_i\pi_r}-z_{\pi_i\pi_s}\right)A_{ri}\left(z_{\pi_s\pi_i}-z_{\pi_r\pi_i}\right)+A_{si}\left(z_{\pi_r\pi_i}-z_{\pi_s\pi_i}\right)\right) \tag{10}$$

式 (10) 的时间复杂度仅为 O(n). 如果 π'是通过交换 r 和 s 后得到的邻域解, 则对增益矩阵 $\delta(\pi', u, v)$ 进行计算从而能更快速得到下一个邻域解的目标函数值, 但必须遵循$\{r,s\}\cup\{u,v\}=\emptyset$的条件, 从而时间复杂度仅为 O(1).

$$\delta(\mu,u,v)=(A_{ru}-A_{rv}+A_{sv}-A_{su})(z_{\mu_s\mu_u}-z_{\mu_s\mu_v}+z_{\mu_r\mu_v}-z_{\mu_r\mu_u})+(A_{ur}-A_{vr}+A_{vs}-A_{us})(z_{\mu_u\mu_s}-z_{\mu_v\mu_s}+z_{\mu_v\mu_r}-z_{\mu_u\mu_r})+\delta(\pi,u,v) \tag{11}$$

式 (11) 在计算增益方面采用前解指导后解的思路, 在保留每一阶段的最优值同时进行置换计算, 大大提高目标函数值计算效率.

BLS 算法终止须满足下列条件之一, 则停止迭代输出最佳解结果: 1) 达到给定最大迭代次数 10 000 次; 2) 达到给定最大计算时间 600 秒.

3 算例分析

为了验证算法的求解质量与效率, 将所提出的算法 BLS 与 Boysen 等[14]提出的束搜索算法 (Beam Search, BS) 和数学规划模型的计算结果进行对比. 数学规划模型求解采用 Gurobi8.0 求解器, 计算时间设定为 1800 秒. BLS 和 BS 算法采用 C#编程并在 Visual Studio 2013 中实现. 所有方法在 CPU 为 Inter(R)Core(TM)i7-4790M@ 3.60 GHz、内存为 8 GB 以及系统为 Windows7 的个人电脑完成测试. 将各个方法求解得到的目标函数值转换成相对百分偏差 (Relative Percent Deviation, *RPD*), 以此作为算法性能分析的响应变量. *RPD* 计算如下式所示:

$$RPD=\frac{f_{\text{best}}-f_{\text{alg}}}{f_{\text{best}}}\times 100\% \tag{12}$$

其中, f_{alg}为指定算法求解某个算例的目标函数值, f_{best} 为所有方法求得的最佳目标值. 由式 (12) 可知, *RPD* 的值越小表明对应的算法求解效果越好.

3.1 算例设计

针对所研究的集装箱班列转运调度问题, 尚未有公开的算例测试集. 在此利用 Boysen 等[14]提出的方法产生本问题的测试算例. 为了生成测试算例, 表 1 中列出了用于产生算例相关参数的取值范围. 矩阵 A_{ij} 产生方式如下: 用车厢数量乘以载荷系数产生班列间能够交互的最大集装箱箱数; 对于每辆班列 j, 随机选取另一辆班列 i, 其能够转运的集装箱箱数从区间[1, *MaxTr*]中采用均匀分布随机产生, 其中 *MaxTr* 是指两辆班列之间能够转运的集装箱箱数上限; 随后, 班列 i 被标记为禁忌状态; 重复上述过程直至达到班列 j 的最大集装箱数或所有班列均为禁忌状态为止.

表 1 算例参数表

参数	描述	数值
m	轨道数	2 4 6 8 10
$\|I\|$	班列数	12 16 24 32 36 48 60 80 100
Wg	车厢数量	20 30 40
LF	载重系数	1.0 1.4 1.8
$MaxTr$	班列间的转运上限	24 20 16 12

算例的产生充分考虑时间限制的生成. 若不限制的话, 则班列 i 设置 e_i=1 和 l_i=T. 若仅有进入时间限制, 则从区间[1, T]随机确定 e_i 并且所有 l_i=T, 其中 e_i=1 有 50% 的概率出现而其余则采用均匀分布随机产生. 若进入与离开均有时间限制, 则从区间[1, T/2] 中选择 e_i, 区间[T/2, T]中选择 l_i, 对于 e_i=1 有 50% 的概率出现, 对于 l_i=T 也有 50% 的概率出现. 在此所有生成的算例都需要试算以保证至少存在一个可行解, 否则将重新生成. 表 2 为算例中班列与轨道的关系.

表 2 班列数与轨道数取值范围

参数	数值			
班列	12,16,24,36,48	12,24,36	16,24,32	60,80,100
轨道	2,4	6	8	10

根据班列的进出堆场的时间段将算例分为 FTSPST1, FTSPST2 和 FTSPST3 三种类型, FTSPST1 代表所有班列最早进入与最晚离开堆场时间段一样; FTSPST2 代表所有班列最早进入堆场时间段不同最晚离开堆场时间段相同; FTSPST3 代表所有班列最早进入最晚离开堆场时间段都不相同. 针对表 2 所示的班列数和轨道数生成 105 组算例, 其中 12-2, 12-4 和 16-2, 16-4 的算例总共生成 5 组, 其余都为 1 组.

基于初始计算测试, BLS 算法采用表 3 所示的参数将获得不错的计算性能, 以求在较短的时间给出不错的求解结果.

表 3 BLS 算法参数表

参数	参数描述	取值
r_1	定向扰动参数	7
r_2	定向扰动参数	3
p_0	定向扰动概率	0.9
Q	随机扰动概率	0.2
Ω	连续未改进解数量的阈值	2500
K_0	初始扰动强度	0.15

3.2 算例计算分析

将班列数在 50 辆以下的归为小规模算例, 而将班列数在 50 辆以上归为大规模算例. 由于篇幅限制, 在此仅列出班列数为 12 和 80 的算例计算详细结果. 表 4 给出了班列数为 12 的一组算例计算结果. 在该表中列出了各个算法求得的目标函数、计算时间以及利用式 (12) 计算出的相对百分偏差 *RPD*, 其中 BLS 与 BS 的计算时间为 600 秒内首次搜索到该算例最佳解时记录的时间. 下标 BLS 表示 BLS 算法相关的结果, 下标 BS 表示束搜索算法相关的结果, 下标 G 表示优化器 Gurobi 的相关结果. 从中可以看出 3 个方法均能够在较短的计算时间内给出最优解, 但 BLS 的计算时间最短, BS 次之, 优化器 Gurobi 耗时最长.

表 4 12 辆班列算例计算结果

算例类型		f_{BLS}	f_G	f_{BS}	T_{BLS}	T_G	T_{BS}	RPD_G	RPD_{BS}
12-2	FTSPST1	77	77	77	0.002	2.912	0.009	0	0
	FTSPST2	77	77	77	0.003	1.002	0.009	0	0
	FTSPST3	74	74	74	0.004	1.402	0.007	0	0
12-4	FTSPST1	126	126	126	0.002	0.236	0.008	0	0
	FTSPST2	93	93	93	0.004	0.144	0.005	0	0
	FTSPST3	98	98	98	0.003	0.437	0.007	0	0
12-6	FTSPST1	126	126	126	0.004	0.072	0.007	0	0
	FTSPST2	90	90	90	0.001	0.010	0.005	0	0
	FTSPST3	100	100	100	0.008	0.048	0.007	0	0

表 5 给出了班列数为 80 时的计算结果, 其中涉及的指标与表 4 一样. 从中可以看出 BLS 依然给出所有算例的最好解, 但其计算时间比 BS 的用时稍长一些. 优化器 Gurobi 耗光了给定的时间限制 1800 秒, 且求解性能易受算例类型的影响. 也就是说进入或离开堆场的时间段的不同使得求解空间的规模不一样, 优化器求解性能亦不一样. BS 的求解质量相对较为稳定, *RPD* 的均值为 13.3%.

表 5 80 辆班列算例计算结果

算例类型		f_{BLS}	f_G	f_{BS}	T_{BLS}	T_G	T_{BS}	RPD_G(%)	RPD_{BS}(%)
80-10	FTSPST1	798	582	710	51.50	1800	30.39	27.1	11.0
	FTSPST2	749	743	627	121.36	1800	18.08	0.8	16.3
	FTSPST3	704	687	615	74.953	1800	10.02	2.4	12.6

表 6 和表 7 分别给出了小规模算例与大规模算例的平均计算结果. 从计算精度上来说, 对于 50 辆以下的班列数, BLS 算法能很快计算出与 Gurobi 相同的最优解, 而 BS 则与最优解存在 1.69% 的差距. 对于 50 辆以上班列数的大规模算例, Gurobi 的平均计算时间为 24 分钟, BLS 算法为 105.59 秒, 而 BS 算法仅为 26.11 秒, 可见在大规模算例上 BS 算法有优势; 但是在解的精确度上 BLS 算法明显比 Gurobi 和 BS 的计算结果更好, 而且 BS 算法在解 50 以上规模的 FTSPST2 和 FTSPST3 类型算例时解的精度较差, 和 Gurobi 的计算结果相比存在较大的差距. 对于 100 辆左右的班列数, Gurobi 优化器和 BS 算法很难求出该问题的较优解, 而 BLS 能够在 230.66 秒的平均计算时间内给出最好的解. 因此, BLS 算法针对集装箱班列的同步转运调度问题具有较为明显的优势, 不仅计算精确度高, 而且计算耗时也很短.

表 6 小规模算例平均计算结果

方法	评估指标		
	AvgT_G (s)	Avg *RPD*(%)	Sum T_G(s)
Gurobi	236.58	0.34	22 711.76
BLS	0.17	-	15.95
BS	0.55	1.69	52.74

表 7 大规模算例平均计算结果

方法	评估指标		
	AvgT_G (s)	Avg *RPD*(%)	Sum T_G(s)
Gurobi	1424.22	11.67	12 818.01
BLS	105.59	-	950.28
BS	26.11	9.92	234.98

4 结论

本文在考虑同步转运作业需求的前提下, 研究了集装箱班列调度问题, 以最大化同步转运的集装箱箱数为目标, 提出了相应的混合整数规划模型. 由于该问题的复杂性, 本文提出利用突破性局部搜索框架对问题进行求解. 该算法依据搜索状态自适应确定扰动类型与扰动幅度, 以突破局部最优解对迭代过程的束缚. 基于问题的结构特征, 提出利用进入离开堆场时间段排序的方式产生初始解, 同时重定义了目标函数值增益矩阵的计算方式. 算例分析表明, 本文所提出的 BLS 算法较之 Gurobi 优化器与文献中的 BS 算法具有明显的求解优势, 且明显提高了调度优化的计算效率. 该优化方法对于铁路集装箱站场提升运作管理效率具有一定的理论指导意义. 未来将公路-铁路联运的转运需求纳入该调度问题中将进一步提升站内作业效率.

参考文献

1 交通运输部. 2017 年交通运输行业发展统计公报. http://www.gov.cn/xinwen/2018-03/30/content_5278569.htm. [2018-03-30].

2 国家发展改革委, 交通运输部, 中国铁路总公司. “十三五”铁路集装箱多式联运发展规划. http://www.gov.cn/xinwen/2017-05/12/content_5193215.htm. [2017-04-19].

3 Boysen N, Fliedner M, Jaehn F, *et al*. A survey on container processing in railway yards. Transportation Science, 2013, 47(3): 312–329. [doi: 10.1287/trsc.1120.0415]

4 Otto A, Pesch E. Operation of shunting yards: Train-to-yard assignment problem. Journal of Business Economics, 2017, 87(4): 465–486. [doi: 10.1007/s11573-016-0827-3]

5 Boysen N, Fliedner M, Kellner M. Determining fixed crane areas in rail-rail transshipment yards. Transportation Research Part E: Logistics and Transportation Review, 2010, 46(6): 1005–1016. [doi: 10.1016/j.tre.2010.05.004]

6 Kellner M, Boysen N, Fliedner M. How to park freight trains on rail-rail transshipment yards: The train location problem. OR Spectrum, 2012, 34(3): 535–561. [doi: 10.1007/s00291-011-0246-3]

7 Fedtke S, Boysen N. Gantry crane and shuttle car scheduling in modern rail-rail transshipment yards. OR Spectrum, 2017, 39(2): 473–503. [doi: 10.1007/s00291-016-0461-z]

8 Fedtke S, Boysen N. A comparison of different container sorting systems in modern rail-rail transshipment yards. Transportation Research Part C: Emerging Technologies, 2017, 82: 63–87. [doi: 10.1016/j.trc.2017.06.012]

9 Guo P, Cheng WM, Zhang ZQ, *et al*. Gantry crane scheduling with interference constraints in railway container terminals. International Journal of Computational Intelligence Systems, 2013, 6(2): 244–260. [doi: 10.1080/18756891.2013.768444]

10 Guo P, Cheng WM, Wang Y, *et al*. Gantry crane scheduling in intermodal rail-road container terminals. International Journal of Production Research, 2018, 56(16): 5419–5436. [doi: 10.1080/00207543.2018.1444812]

11 王力, 朱晓宁, 谢征宇, 等. 基于混堆的铁路集装箱中心站堆场箱位指派模型. 中南大学学报 (自然科学版), 2013, 44(S1): 271–275.

12 王力, 朱晓宁, 闫伟, 等. 铁路集装箱中心站堆场混堆优化模型. 交通运输系统工程与信息, 2013, 13(2): 172–178. [doi: 10.3969/j.issn.1009-6744.2013.02.026]

13 Boysen N, Emde S, Stephan K, *et al*. Synchronization in hub terminals with the circular arrangement problem. Naval Research Logistics, 2015, 62(6): 454–469. [doi: 10.1002/nav.21640]

14 Boysen N, Jaehn F, Pesch E. Scheduling freight trains in rail-rail transshipment yards. Transportation Science, 2011, 45(2): 199–211. [doi: 10.1287/trsc.1100.0365]

15 Benlic U, Hao JK. Breakout local search for the quadratic assignment problem. Applied Mathematics and Computation, 2013, 219(9): 4800–4815. [doi: 10.1016/j.amc.2012.10.106]

16 Benlic U, Hao JK. Breakout Local Search for maximum clique problems. Computers & Operations Research, 2013, 40(1): 192–206.

17 Benlic U, Hao JK. Breakout local search for the max-cutproblem. Engineering Applications of Artificial Intelligence, 2013, 26(3): 1162–1173. [doi: 10.1016/j.engappai.2012.09.001]

18 Ghandi S, Masehian E. A breakout local search (BLS) method for solving the assembly sequence planning problem. Engineering Applications of Artificial Intelligence, 2015, 39: 245–266. [doi: 10.1016/j.engappai.2014.12.009]

19 Benlic U, Burke EK, Woodward JR. Breakout local search for the multi-objective gate allocation problem. Computers & Operations Research, 2017, 78: 80–93.