中国科学: 信息科学 2022 年 第 52 卷 第 11 期: 2121–2140
SCIENTIA SINICA Informationis
论文

车站封锁下基于问题知识的高速铁路列车运行实时调整方法

王荣笙[1,2,4], 张琦[2,4*], 张涛[2,4], 林鹏[3,4], 丁舒忻[2,4], 袁志明[2,4]

1. 中国铁道科学研究院研究生部, 北京 100081
2. 中国铁道科学研究院集团有限公司通信信号研究所, 北京 100081
3. 中南大学自动化学院, 长沙 410083
4. 中国铁道科学研究院集团有限公司国家铁路智能运输系统工程技术研究中心, 北京 100081
* 通信作者. E-mail: zhangqi@rails.cn

收稿日期: 2021–09–22; 修回日期: 2021–11–08; 接受日期: 2022–01–24; 网络出版日期: 2022–11–10

国家自然科学基金 (批准号: U1834211, 61790575, U1934220)、中国国家铁路集团有限公司科技研究开发计划课题 (批准号: K2021X001) 和中国铁道科学研究院集团有限公司科研项目 (批准号: 2021YJ043) 资助

摘要 针对突发事件导致的车站封锁情况, 本文以列车运行图为问题对象, 以进化计算框架为基础, 提出基于问题知识的运行图实时调整方法, 通过减小列车总晚点时间, 保证高铁运营的安全高效和旅客的满意舒适. 首先, 基于调整列车发车次序的运行图调整策略提出排列编码方法, 用于减少解空间的无效搜索. 之后, 根据 "紧追踪" 的列车运行追踪方式, 设计启发式解码方法消除所有行车作业约束, 提升算法求解效率. 最后, 将调度员调整运行图的经验作为问题知识, 用于初始化进化计算的初始种群, 由此提出基于问题知识的启发式种群初始化方法, 加快算法前期的收敛速度并提高求解方案质量. 以京津高速线为例, 在北京南站设置车站封锁下 20~150 min 不同封锁时长的 9 个典型场景, 选择加强精英保留遗传算法和差分进化算法, 分别应用实整数编码和排列编码, 与随机种群初始化和启发式种群初始化的不同组合进行仿真实验. 仿真结果表明, 相较于实整数编码难以获取可行解, 2 种进化算法应用排列编码方法后, 能在 9 s 的平均时间内给出列车总晚点时间最小的调整方案. 在启发式种群初始化的改进下, 2 种进化算法能更快地收敛于近似最优解. 选取加强精英保留遗传算法应用排列编码和启发式种群初始化的改进变体, 作为本文最优改进进化算法. 针对 CPLEX 无法在 10 min 获得最优解的 7 个场景, 该改进进化算法都能在 20 s 内给出近似最优解.

关键词 高速铁路, 列车运行调整, 车站封锁, 进化计算, 遗传算法, 排列编码优化

引用格式: 王荣笙, 张琦, 张涛, 等. 车站封锁下基于问题知识的高速铁路列车运行实时调整方法. 中国科学: 信息科学, 2022, 52: 2121–2140, doi: 10.1360/SSI-2021-0332
Wang R S, Zhang Q, Zhang T, et al. Real-time rescheduling approach of train operation for high-speed railways using problem-specific knowledge under a station blockage (in Chinese). Sci Sin Inform, 2022, 52: 2121–2140, doi: 10.1360/SSI-2021-0332

1 引言

高速铁路在推进经济建设、加快碳中和进程、促进城市协调发展和服务旅客运营等方面提供重要的支撑和引领作用. 我国高速铁路已逐渐形成 “八纵八横” 的复杂路网, 但高铁运营过程中面临自然灾害、设备故障、人为决策失误等不同突发事件的影响, 列车易出现大范围延误, 影响旅客出行. 当突发事件影响列车运行较为严重时, 调度员需要及时下达车站封锁、区间封锁或者区间临时限速等调度命令, 避免突发事件危及行车安全. 同时, 调度员凭自身经验在调度指挥系统中快速准确地下达阶段计划, 工作强度较大, 调整方案的实时性和最优性难以同时保证.

作为一类 NP 难问题 [1], 列车运行调整通过优化列车在各站的接发车时刻, 运行进路和发车次序 [2], 尽可能恢复正常行车秩序. 列车运行调整的求解方法主要包括运筹学、人工智能方法和进化计算. 运筹学方法下的基本模型包括整数规划模型和替代图和车站作业调度模型. 整数规划模型考虑列车在车站和区间的作业约束, 以列车总晚点时间 [3]、取消列车数量 [4] 和旅客旅行时间 [5] 为优化目标建立模型. 替代图在检测和消解列车运行冲突的问题中应用较多 [6]. 车站作业调度模型常与替代图结合, 并将列车运行调整过程近似为零部件加工过程 [7], 考虑如何优化零部件加工顺序和加工时间使总加工时间最小, 即如何调整列车运行顺序和各列车占用闭塞分区或股道的时间, 使列车总晚点时间最小. 由此可见, 列车对应于零部件, 闭塞分区及股道相当于加工机器. 综上, 运筹学求解列车运行调整问题的基本思路是从数学规划角度建立模型, 基于商业优化软件 CPLEX 或者 GROUBI 求解列车运行调整最优方案. 但当面临强耦合、非线性、多时变的列车运行时空约束时, 调整方案的实时性求解存在较大挑战. 随着人工智能技术在智能交通中的发展 [8], 以强化学习为代表的人工智能方法在列车运行调整问题中广泛研究 [9~11], 训练后的离线训练模型可直接用于同类型列车运行调整问题的实时求解. 例如, 部分学者通过建立列车运行调整的强化学习环境, 基于蒙特卡洛 (Monte Carlo) 树搜索 [12] 和 Deep Q Network [13] 的强化学习方法, 实现列车总晚点时间最小的最优调整方案. 但调整方案是否最优比较依赖于强化学习环境的设计和前期较优策略的搜索 [12, 13]. 进化计算利用种群基因的选择、交叉和变异等特性解决调度和优化问题 [14]. 虽然难以保证每次调整方案的最优性, 但可在更短的时间内给出一个近似最优解. 以蚁群算法 [15]、遗传算法 [16] 和粒子群算法 [17] 为代表的进化算法已解决大量的列车运行调整问题.

综上, 大量研究集中于如何提升列车运行调整方案的实时性和最优性. 然而, 实际列车运行调整过程中, 调度员每次所考虑的首要指标并不一定是调整方案的最优性, 更重要的是根据列车运行过程中的复杂时空约束, 如何给出切实可行且实时性较高的列车运行调整方案. 在以上求解方法中, 与人工智能方法相比, 进化计算利用种群基因特性能搜索出相对更优的策略. 与运筹学方法相比, 进化计算求解列车运行调整问题的实时性更高, 能在有限时间内给出近似最优的列车运行调整方案. 但仅使用进化计算可能导致某些调整方案的实时性和最优性较差. 在其他领域的调度优化问题中, 研究学者针对进化计算的上述局限性, 在进化计算求解过程中加入特定的问题知识和启发式规则 [18, 19], 以提升调整方案的求解质量和计算效率. 目前, 在列车运行调整问题中, 较少有研究将问题知识与进化算法相结合, 用于提升进化计算求解效率. 因此, 本文针对车站封锁情况, 兼顾列车运行调整方案最优性和实时性的同时, 提出基于问题知识的列车运行实时调整方法, 通过不同进化算法在典型场景上的对比实验, 验证所提出方法的可行性和有效性. 主要创新点总结如下: (1) 针对车站封锁情况, 基于调整列车发车次序的运行图调整策略, 提出发车次序的排列编码方法. 与既有研究大多采用接发车时刻的实整数编码方法相比, 本文所提出的排列编码方法可有效减少解空间的无效搜索; (2) 针对列车运行调整问题中的复杂时空约束, 基于 “紧追踪” 的列车运行追踪方式设计启发式解码方法, 将列车运行调整问题转

化为无约束优化问题, 提升进化算法的搜索效率; (3) 将调度员调整运行图的经验设置为问题知识, 提出基于问题知识的启发式种群初始化方法, 保证种群多样性的前提下, 提升算法前期收敛速度和求解方案质量.

其余章节结构如下: 第 2 节介绍列车运行调整问题和车站封锁情况下的假设条件; 第 3 节建立车站封锁情况下的列车运行调整模型; 第 4 节针对列车运行调整的问题特点和优化目标, 阐述基于问题知识的运行图实时调整方法的具体步骤, 详细介绍编码、解码和种群初始化在进化计算框架上的具体改进; 第 5 节针对本文所提出方法下的改进进化算法进行实验验证和结果对比分析; 第 6 节总结和展望本文工作.

2 问题描述

本文针对车站封锁情况下的列车运行调整问题, 遵循我国高速铁路所规定的技术条件和规章制度, 分析突发事件下的车站封锁情况并给出相应的假设条件. 考虑的车站封锁是指受突发事件影响, 调度员需要对车站实施封锁命令, 致使大量列车在封锁车站临时待避, 待封锁解除后方可发车. 需要注意的是, 封锁车站可进行接车作业, 但不能发送列车. 根据最新《智能调度集中系统暂行技术条件》[20], 车站封锁情况下, 列车在封锁车站后续车站的运行顺序不可改变, 故只能优化受影响列车在封锁车站的发车次序. 考虑高铁线路上列车和车站的数量分别为 I 和 J, 则列车集合和车站集合分别表示为 $\mathbb{I}=\{1,2,\ldots,I\}$, $\mathbb{J}=\{1,2,\ldots,J\}$. 列车 i ($i\in\mathbb{I}$) 从始发站 1 站出发, 在沿途中间站 j ($j\in\mathbb{J}$) 通过或者停站, 最终到达终点站 J. 令车站 j' ($j'\in\mathbb{J}$) 为封锁车站, t_{start} 和 t_{end} 分别表示封锁的开始时刻和结束时刻, 则车站封锁的时间范围为 $[t_{\text{start}},t_{\text{end}})$, 即所有受影响列车在 t_{end} 时间点之后可从封锁车站 j' 发车.

本文并未考虑区间封锁条件, 原因如下. 按照《智能调度集中系统暂行技术条件》[20] 规定, 发生区间封锁时, 列车运行顺序同样不可改变. 故区间封锁仅导致受影响列车按照基本运行图所规定的运行顺序晚点到达后续车站, 实际仅需要调整列车在区间的追踪列车间隔时间和在车站的停站时间, 基本不会涉及发车次序的调整. 同样, 本文不考虑接发车作业均失效的车站封锁, 其原因是, 当封锁车站不再接发列车时, 列车也只能按照原有列车运行顺序, 在封锁车站前方多个车站待避. 待封锁命令解除后, 列车以原有运行顺序通过封锁车站.

针对本文所考虑的车站封锁情况, 做出如下假设: (1) 列车在沿线各站不可提前到站或发车, 以保证列车在车站安全地接发车作业, 以及旅客在车站的正常上下车; (2) 列车运行调整过程中, 车站封锁的开始时刻 t_{start} 和结束时刻 t_{end} 不可改变; (3) 依据我国《铁路技术管理规程》(高速铁路部分) 第五章第 140 条, 车站可与动车段通过走行线相连接, 本文令封锁车站 j' 附近至少存在一个动车段, 保证受车站封锁影响的列车可存放于封锁车站及其附近的动车段.

3 列车运行调整建模

3.1 目标函数

实际铁路生产运营过程中, 列车阶段计划是以时间和空间关系的运行图来表示的. 调度员按照调度指挥系统每日所下达的基本运行图, 监督和调整列车的接发车时刻. 受车站封锁影响, 调度员调整运行图所考虑的基本目标是尽可能减少晚点列车在基本运行图上的偏离. 故设置列车运行实时调整的优

化目标为列车总晚点时间, 即

$$\min \sum_{i=1}^{I}\sum_{j=1}^{J}\left[\left(d_{i,j}-\bar{d}_{i,j}\right)+\left(f_{i,j}-\bar{f}_{i,j}\right)\right],\quad i\in\mathbb{I},\ j\in\mathbb{J}, \tag{1}$$

其中, $\bar{d}_{i,j}$ 和 $\bar{f}_{i,j}$ 分别表示基本运行图中列车 i 在当前车站 j 的图定到站时刻和图定发车时刻, $d_{i,j}$ 和 $f_{i,j}$ 分别表示调整阶段计划过程中, 列车 i 在当前车站 j 的实际到站时刻和实际发车时刻. 由假设 (1) 可知, $d_{i,j}\geqslant\bar{d}_{i,j}$, $f_{i,j}\geqslant\bar{f}_{i,j}$, 故式 (1) 计算的列车总晚点时间是非负的.

3.2 约束条件

列车在高铁线路实际运行时, 具体行车作业包括列车在车站的到站、停站或者通过, 以及发车等. 为方便建模和处理约束, 且不失问题一般性, 将列车的通过作业拆解为作业时刻相等的到站作业和发车作业. 故基本的行车作业约束包括如下几方面.

• 停站时间约束:

$$f_{i,j}-d_{i,j}\geqslant c_{i,j}^{\min},\quad i\in\mathbb{I},\ j\in\mathbb{J}, \tag{2}$$

其中, $c_{i,j}^{\min}$ 表示列车 i 在当前车站 j 的最小停站时间. 式 (2) 表示不应压缩列车原有的停站时间, 以保证旅客的安全乘降.

• 区间运行时间约束:

$$d_{i,j+1}-f_{i,j}\geqslant b_{i,j}^{\min},\quad i\in\mathbb{I},\ j,j+1\in\mathbb{J}, \tag{3}$$

其中, $d_{i,j+1}$ 表示列车 i 在下一车站 $j+1$ 的到站时刻, $f_{i,j}$ 表示列车 i 在当前车站 j 的发车时刻, $b_{i,j}^{\min}$ 表示列车 i 在区间 $(j,j+1)$ (车站 j 和 $j+1$ 之间的站间区间) 的最小区间运行时间. 式 (3) 表示受列车牵引制动性能限制, 列车的实际区间运行时间应满足最小值约束.

• 相邻两列车的到达或者发车间隔时间约束:

$$d_{i+1,j}-d_{i,j}\geqslant\tau_{\mathrm{dd}},\quad i,\ i+1\in\mathbb{I},\ j\in\mathbb{J}, \tag{4}$$

$$f_{i+1,j}-f_{i,j}\geqslant\tau_{\mathrm{ff}},\quad i,\ i+1\in\mathbb{I},\ j\in\mathbb{J}, \tag{5}$$

其中, $d_{i+1,j}$ 和 $f_{i+1,j}$ 分别表示列车 $i+1$ 在当前车站 j 的实际到站时刻和实际发车时刻, τ_{dd} 和 τ_{ff} 分别为基本运行图所规定的最小到达间隔时间和最小发车间隔时间. 式 (4) 和 (5) 分别表示相邻两列车 i 和 $i+1$ 在当前车站 j 的到站作业之间和发车作业之间分别至少间隔 τ_{dd} 和 τ_{ff}.

车站封锁情况下, 首先需要确定线路上受影响的具体列车, 再确定上述列车从封锁车站可发车的时刻. 设 $\bar{i}$ 和 i'' 分别为不受车站封锁影响和受封锁影响的列车. 车站封锁结束后, 令 i' 为从封锁车站 j' 发出的第一列列车的序号.

• 列车在封锁车站的发车时刻约束

$$\begin{cases} f_{\bar{i},j'}=\bar{f}_{\bar{i},j'}, & \bar{f}_{\bar{i},j'}<t_{\mathrm{start}},\ \bar{i}\in\mathbb{I}_{\mathrm{no}},\\ f_{i',j'}=t_{\mathrm{end}}, & \bar{f}_{i',j'}>t_{\mathrm{start}},\\ f_{i'',j'}=\bar{f}_{i'',j'}+\tau_{\mathrm{dd}}, & \bar{f}_{i'',j'}>t_{\mathrm{start}},\ i''\in\mathbb{I}_{\mathrm{blg}}, \end{cases} \tag{6}$$

其中, $\mathbb{I}_{\mathrm{no}}=\{1,2,\ldots,i'-1\}$ 和 $\mathbb{I}_{\mathrm{blg}}=\{i'+1,i'+2,\ldots,I\}$ 分别表示不受车站封锁影响和受车站封锁影响 (不包括列车 i') 的列车集合. 式 (6) 中, 由于不受封锁影响, 列车 $\bar{i}$ 在封锁车站的图定发车时刻 $\bar{f}_{\bar{i},j'}$

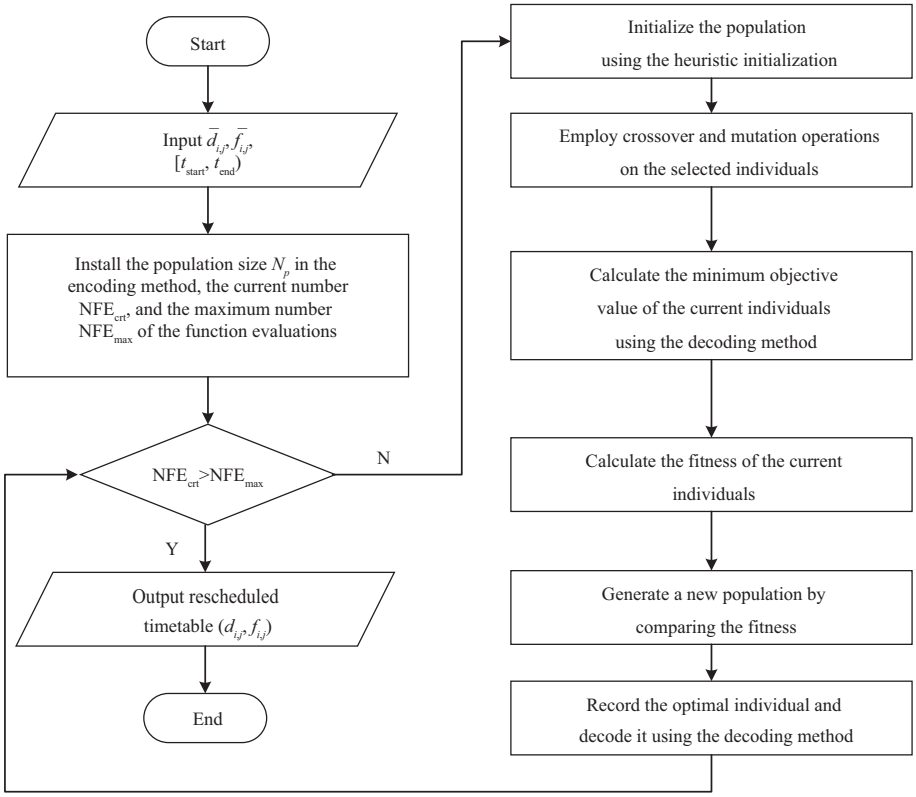

图 1 基于问题知识的运行图实时调整方法

Figure 1 Real-time train rescheduling approach based on problem-specific knowledge

小于封锁的开始时刻 t_{start}, 则列车 $\bar{i}$ 实际按照图定发车时刻 $\bar{f}_{\bar{i},j'}$ 正点发车. 由于列车 i' 是从封锁车站 j' 出发的第一列列车, 故列车 i' 的实际发车时刻 $f_{i',j'}$ 等于封锁的结束时刻 t_{end}. 其余受封锁影响的列车 i'' 的实际发车时刻 $f_{i'',j'}$ 等于列车 i'' 的图定发车时刻 $\bar{f}_{i'',j'}$ 与最小发车间隔时间 τ_{dd} 之和.

4 基于问题知识的运行图实时调整方法

由第 3 节列车运行调整模型可知, 车站封锁情况下, 随着所考虑列车数量的增多, 问题求解难度呈现指数级增长. 由此可见, 列车运行调整问题属于 NP 难问题. 针对此类问题, 进化计算可在有限时间内给出近似最优解. 但仅使用进化计算可能导致某些调整方案的求解质量和效率较低, 故需要在进化计算框架上设置特定的问题知识和启发式规则, 提升调整方案的实时性和最优性. 为此, 本文针对车站封锁影响下的高速铁路列车运行调整问题, 基于进化计算框架提出基于问题知识的运行图实时调整方法, 具体步骤如图 1 所示.

首先, 输入基本运行图的图定到站时刻 $\bar{d}_{i,j}$, 图定发车时刻 $\bar{f}_{i,j}$ 和车站封锁时间范围 $[t_{\text{start}}, t_{\text{end}})$. 初始化进化计算基本参数, 包括所提出编码方法下的种群规模大小 N_p, 适应度函数评价的当前次数 NFE_{crt} 和最大次数 $\text{NFE}_{\max}$, 以及所提出种群初始化方法下的初始种群. 之后, 检测当前适应度函数的评价次数是否达到其最大值. 若是, 则直接输出封锁车站当前发车次序下的实绩运行图; 否则, 对所选择种群个体进行选择、交叉和变异的操作, 计算所提出解码方法下当前种群个体的最优目标函数值 (即最小列车总晚点时间), 通过适应度评价函数计算种群个体的适应度大小, 选择出最优种群个体

并对其解码. 最后, 根据进化计算求得的受影响列车在封锁车站的发车次序, 生成实绩运行图 (即 $d_{i,j}$, $f_{i,j}$). 结合以上步骤, 基于问题知识的运行图实时调整方法的伪代码如算法 1 所示. 综上, 本文所提出基于问题知识的运行图实时调整方法主要包括: (1) 基于调整发车次序的问题目标提出排列编码方法; (2) 基于 "紧追踪" 的列车运行追踪方式设计启发式解码方法; (3) 利用调度员调图经验提出启发式初始化方法. 以下详细阐述所提出方法下编码、解码和种群初始化在进化计算框架上的具体改进.

Algorithm 1 Real-time train rescheduling algorithm based on problem-specific knowledge

Input: The planned arrival and departure time $\bar{d}_{i,j}$, $\bar{f}_{i,j}$, the blocked window $[t_{\text{start}}, t_{\text{end}})$.

Output: The rescheduled timetable $d_{i,j}$, $f_{i,j}$.

```
Install N_pop, NFE_max;
Install the initial population pop as train departure sequences using Algorithm 3;
NFE_crt ⇐ 1;
while NFE_crt ≠ NFE_max do
    Employ crossover and mutation operations on the selected individuals;
    Obtain the train timetable of the current population using Algorithm 2;
    Calculate the objective values and generate a new population;
    NFE_crt ⇐ NFE_crt + 1;
end while
Record the optimal individual and decode it using Algorithm 2;
```

Return: $d_{i,j}$, $f_{i,j}$.

4.1 基于发车次序的排列编码

进化计算框架下编码方法决定种群染色体中个体的维度, 进而影响算法解空间大小和问题求解效率. 运行图的调整对象通常为实际接发车时刻, 故针对列车运行调整问题, 大多数研究学者采用基于列车接发车时刻的实整数编码方式 [21, 22]. 基于第 3 节所述模型, 列车运行调整的实整数编码方法表示为

$$\boldsymbol{E}_{\text{RI}} = [d_{1,1}, f_{1,1}, \ldots, d_{i,j'}, f_{i,j'}, \ldots, d_{I,J}, f_{I,J}], \quad i \in \mathbb{I},\ j' \in \mathbb{J}, \tag{7}$$

其中, $\boldsymbol{E}_{\text{RI}}$ 表示实整数编码下进化算法种群染色体的其中一个个体, 其向量维度为 $2 \cdot I \cdot J$. 由于列车 i 在始发站 1 站和终点站 J 分别不存在到站时刻 $d_{1,1}$ 和发车时刻 $f_{I,J}$, 为方便处理问题且不失问题一般性, 令 $d_{1,1} = f_{1,1}$, $f_{I,J} = d_{I,J}$. $\boldsymbol{E}_{\text{RI}}$ 作为种群染色体的其中一个个体, 在进化计算选择、交叉和变异的操作下, 调整运行图的接发车时刻和发车次序.

由式 (7) 可知, 实整数编码方法存在比较严重的 "维度灾难" 问题, 即随着列车运行调整模型所考虑列车和车站的数量增加, 问题的求解难度呈现指数级增长. 为此, 本文针对车站封锁情况, 调整列车在封锁车站的发车次序使列车总晚点时间最小, 由此提出基于发车次序的排列编码方法, 即

$$\boldsymbol{E}_{\text{PM}} = [o_{1,j'}, o_{2,j'}, \ldots, o_{i,j'}, \ldots, o_{I,j'}], \quad i \in \mathbb{I},\ j' \in \mathbb{J}, \tag{8}$$

其中, 排列编码方法下, 进化算法种群染色体的其中一个个体设定为所有列车在封锁车站 j' 的发车次序 ($o_{1,j'}$, $o_{2,j'}$, ..., $o_{i,j'}$, ..., $o_{I,j'}$). 假设调整一天 24 h (1440 min) 的运行图, 则实整数编码下解空间为运行图整个时间域, 即 $1440^{2 \cdot I \cdot J}$, 排列编码下解空间为列车在封锁车站的发车次序种类 (即 $I!$). 一般情况下, 高铁调度区段 3 ~ 4 h 阶段计划内的列车个数 I 不超过 35 列. 故在此前提下, 排列编码下的解空间远小于实整数编码下的解空间. 排列编码方法下, 不同发车次序表示下的不同个体形成完整的

种群, 种群个体之间不断选择、交叉和变异, 直到达到最大函数评价次数 $\mathrm{NFE}_{\max}$ 时, 输出列车在封锁车站的发车次序.

4.2 启发式解码

进化算法在处理行车作业约束 (式 (2)∼(6)) 时, 需要消耗大量计算时间. 对此, 本文在排列编码方法输出列车在封锁车站发车次序的基础上, 按照 "紧追踪" [23] 的列车运行追踪方式, 提出基于问题约束的启发式解码方法, 即通过设计启发式规则消除问题求解过程中的所有行车作业约束, 将所求问题转化为无约束优化问题, 提升算法求解效率. 其中, "紧追踪" 的列车运行追踪方式是指列车在封锁车站后续所有车站按最小停站时间停站, 在后续所有站间区间按最小区间运行时间运行, 并且相邻两列列车按照最小追踪列车间隔时间运行 [23]. 在应用启发式解码方法之前, 首先根据进化算法所求列车在封锁车站的发车次序, 获取列车在所有车站的图定接发车时刻 $\bar{d}_{g,i}^{\mathrm{PM}}$ 和 $\bar{f}_{g,i}^{\mathrm{PM}}$, 基于问题约束的启发式解码方法包括 3 个步骤.

Step 1. 检测所有受影响列车在封锁车站 j' 及其后续车站是否满足发车间隔时间约束 (式 (5)), 若不满足, 则调整受影响列车在上述各站的接发车时刻满足该约束.

Step 2. 按照 "紧追踪" 方式, 后车需要与前车满足到站间隔时间约束 (式 (4)), 故后车在当前车站的实际到站时刻等于前车在该站的实际到站时刻与上述两列车的最小到达间隔时间之和, 由此消除区间运行时间约束 (式 (3)).

Step 3. 后车在当前车站的实际发车时刻一方面由相邻两列车的最小发车间隔时间决定, 另一方面需要考虑后车在当前车站原本的图定停站时间, 综合以上两点考虑消除停站时间约束 (式 (2)).

列车在封锁车站的发车时刻约束 (式 (6)) 不在进化算法的处理流程中, 不会消耗过多求解时间. 综上, 所有行车作业约束被消除. 根据上述分析, 基于问题约束的启发式解码方法的伪代码如算法 2 所示, 其时间复杂度为 $O(I \cdot J)$, 其中, I 和 J 分别为线路上列车和车站的数量. 基于算法 2 输出每一代种群个体经过适应度函数评价后的目标函数. 其次, 进化计算结束后, 根据最终发车次序和算法 2 解码输出实绩运行图.

4.3 基于问题知识的启发式种群初始化

进化计算通常采用随机种群初始化的方法, 为种群染色体提供不同类型的个体. 虽然随机种群初始化保证了种群多样性, 但由于初始种群中缺少较优个体, 进化计算前期搜索效率较低. 为此, 本文提出基于问题知识的启发式种群初始化方法. 设计的问题知识为初始种群提供一个较优的个体, 保证进化计算种群多样性的同时, 提升算法前期的收敛速度和求解方案的质量.

面对列车运行调整问题实际的需求、场景和目标, 不同调度员有各自针对性的问题知识, 问题知识能完整有效地处理车站封锁下的列车运行调整问题. 例如, 按照技术条件和规章制度, 监控列车赶点运行, 或者调整列车在封锁车站的发车次序等, 使受影响列车总晚点最小或者列车晚点数量最少. 本文针对我国高速铁路路情和车站封锁情况, 设计高效的列车运行调整问题知识. 我国高速铁路线路上, 不同列车在各车站的停站次数和停站时间不尽相同, 导致列车从始发站运行至终点站的总运行时间也有所差异, 线路上由此产生不同运行等级的列车. 因此, 设计的问题知识 (problem-specific knowledge, PSK) 如下: 受影响列车在沿线各车站和各站间区间的总运行时间越小, 列车运行等级越高, 在封锁车站更早发车. 基于问题知识 PSK 的启发式种群初始化方法的具体步骤如图 2 所示. 首先, 针对图定发车时刻在 $[t_{\mathrm{start}}, t_{\mathrm{end}})$ 时间范围内的受影响列车 (其集合为 $\mathbb{I}_{\mathrm{PSK}}$), 生成其在 PSK 方法下的发车次序 O_{PSK}. 其次, 针对图定发车时刻不在 $[t_{\mathrm{start}}, t_{\mathrm{end}})$ 时间范围内的受影响列车 (其集合为 $\mathbb{I}_{\mathrm{FSFS}}$), 采用

Algorithm 2 Heuristic decoding algorithm

Input: The planned arrival and departure time $\bar{d}_{i,j}$, $\bar{f}_{i,j}$, the corresponding time $\bar{d}_{g,i}^{\mathrm{PM}}$, $\bar{f}_{g,i}^{\mathrm{PM}}$ in the current population.
Output: The rescheduled timetable $d_{i,j}$, $f_{i,j}$.

$i \Leftarrow 1$;
while $i \neq I$ **do**
  $j \Leftarrow j'$;
  while $j \neq J+1$ **do**
    if $f_{i+1,j'} - f_{i,j'} < \tau_{\mathrm{dd}}$ **then**
      $f_{i+1,j} = f_{i,j} + \tau_{\mathrm{dd}}$;
    end if
    $j \Leftarrow j+1$;
  end while
  $i \Leftarrow i+1$;
end while
$i \Leftarrow 1$;
while $i \neq I+1$ **do**
  $j \Leftarrow j'+1$;
  while $j \neq J+1$ **do**
    $d_{i,j} \Leftarrow \bar{d}_{i,j}^{\mathrm{PM}} + (f_{i,j'} - \bar{f}_{i,j'}^{\mathrm{PM}})$; $f_{i,j} \Leftarrow \bar{f}_{i,j}^{\mathrm{PM}} + (f_{i,j'} - \bar{f}_{i,j'}^{\mathrm{PM}})$; $j \Leftarrow j+1$;
  end while
  $i \Leftarrow i+1$;
end while
$i \Leftarrow 1$, $j \Leftarrow 1$;
while $i \neq I$ **do**
  while $j \neq J+1$ **do**
    if $d_{i+1,j} - d_{i,j} < \tau_{\mathrm{dd}}$ **then**
      $d_{i+1,j} \Leftarrow d_{i,j} + \tau_{\mathrm{dd}}$;
    end if
    if $f_{i+1,j} - f_{i,j} < \tau_{\mathrm{ff}}$ **then**
      $f_{i+1,j} \Leftarrow \max\{f_{i,j} + \tau_{\mathrm{ff}}, d_{i+1,j} + (\bar{f}_{i+1,j}^{\mathrm{PM}} - \bar{d}_{i+1,j}^{\mathrm{PM}})\}$;
    end if
    $j \Leftarrow j+1$;
  end while
  $i \Leftarrow i+1$;
end while
Return: $d_{i,j}$, $f_{i,j}$.

先调度先服务 (first-scheduled-first-served, FSFS) [7] 方法生成列车集合 $\mathbb{I}_{\mathrm{FSFS}}$ 的图定发车次序 O_{FSFS}. 之后, 拼接发车次序 O_{PSK} 与 O_{FSFS}, 生成所有受影响列车在封锁车站完整的发车次序 O_{ALL}. 最后, 将发车次序 O_{ALL} 作为较优个体解, 输入到初始种群中. 根据上述步骤, 启发式种群初始化的伪代码如算法 3 所示. 其中, 受影响列车总运行时间的排序算法 (算法 3 第 6 行的 sort algorithm) 采用 Python 的 sorted 函数, 最坏情况下 sorted 函数的时间复杂度为 $O(I' \cdot \log I')$, 故最坏情况下, 算法 3 的时间复杂度为 $O(I \cdot I' \cdot \log I')$, 其中, I 表示线路上列车数量, I' 表示车站封锁时间范围 $[t_{\mathrm{start}}, t_{\mathrm{end}})$ 内受影响列车的数量.

综上, 算法 1 时间复杂度的计算过程如下. 算法 1 第 5 行选择种群个体的时间复杂度为 $O(\mathrm{NFE}_{\max} \cdot N_p)$, 个体交叉和变异操作的时间复杂度为 $O(\mathrm{NFE}_{\max} \cdot N_p \cdot I)$. 其中, $\mathrm{NFE}_{\max}$ 表示适应度函数的最大评价次数, N_p 表示种群规模大小. 结合算法 2 和 3 的伪代码和时间复杂度, 推导出算法 1 的时间复杂

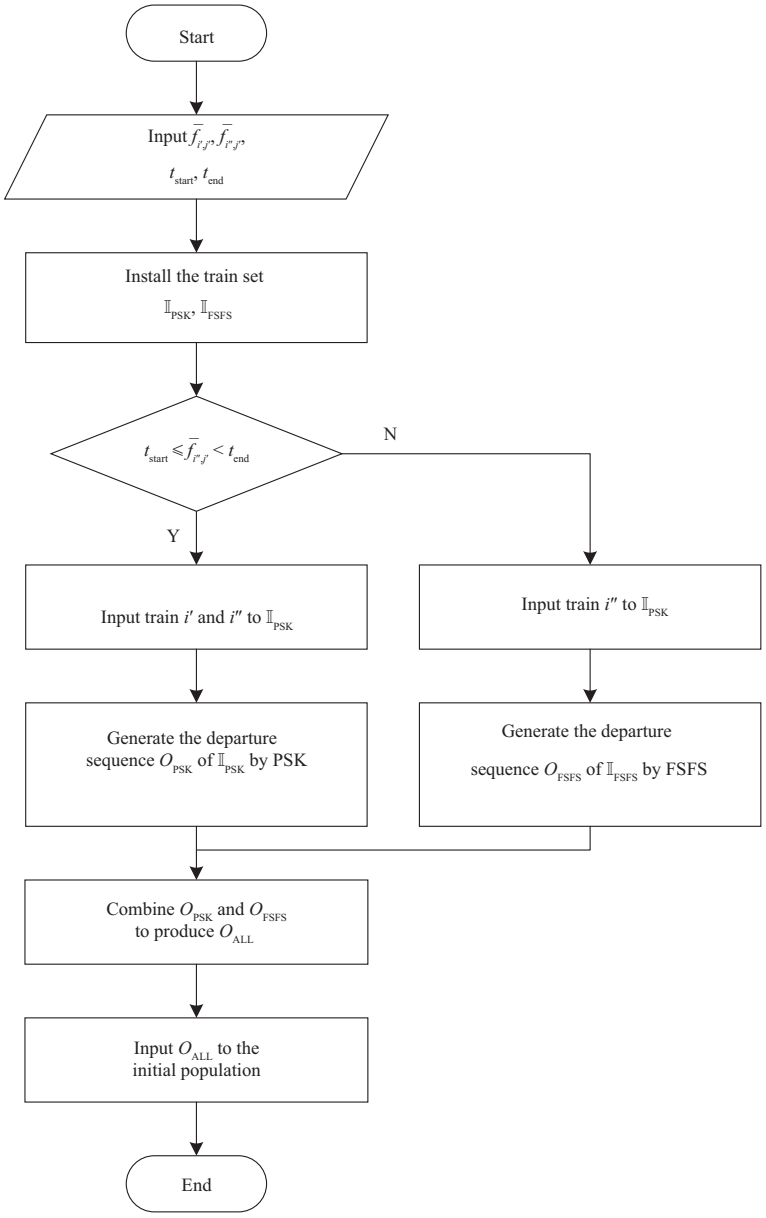

图 2 基于问题知识的启发式种群初始化方法

Figure 2 Heuristic population initialization method based on problem-specific knowledge

度为 $O(I \cdot (\mathrm{NFE}_{\max} \cdot N_p \cdot J + I' \cdot \log I'))$, 其中, J 表示线路上的车站数量.

5 仿真实验

为验证本文所提出基于问题知识的运行图实时调整方法的有效性和可行性, 基于京津高速线设置车站封锁情况下的典型场景, 进行仿真实验. 首先, 通过实验验证, 相较于接发车时刻实整数编码, 本文所提出发车次序排列编码的优越性. 其次, 验证启发式种群初始化方法下改进进化算法的性能提升. 最后, 比较算法性能选出最优改进进化算法, 并与 CPLEX、先调度先服务 FSFS [7] 和问题知

Algorithm 3 Heuristic population initialization algorithm based on PSK

Input: The departure time $\bar{f}_{i',j'}$, $\bar{f}_{i'',j'}$ for train i' and i'' at the blockage station j', the blocked window $[t_{\text{start}}, t_{\text{end}})$.
Output: The completed departure sequence O_{ALL} of the influenced trains.

```
Install the train set I_PSK and I_FSFS;
i'' ⇐ i' + 1;
while i'' ≠ I + 1 do
    if t_start ⩽ f̄_{i'',j'} < t_end then
        Input train i' and i'' to I_PSK;
        Generate O_PSK of I_PSK using PSK and a sort algorithm;
    else
        Input train i'' to I_PSK;
        Generate O_FSFS of I_FSFS using FSFS;
    end if
    i'' ⇐ i'' + 1;
end while
Combine O_PSK and O_FSFS to produce O_ALL, and input O_ALL to the initial population;
```

Return: O_{ALL}.

识 PSK 的 3 种方法进行对比, 验证最优改进进化算法在减小列车总晚点时间上的有效性和实时性. 本文基于 Python 3.6.5 编写列车运行调整模型, 仿真实验的台式计算机配置如下: Intel Core i7-9700T CPU@2.00 GHz, 16 GB 内存.

5.1 典型场景

为验证不同种群染色体编码方法和不同种群初始化方法下改进进化算法的性能提升, 选取京津高速线 2020 年某日下行线列车的基本运行图, 设置车站封锁情况下的典型场景. 京津高速线全线所有 6 个车站的名称及公里标分别为北京南 (0 km)、亦庄 (22 km)、永乐 (46 km)、武清 (84 km)、南仓 (108 km) 和天津 (120 km). 其中, 南仓站是京津高速线上用于划分站间区间的线路所, 不办理客运业务. 5 个站间区间的最小区间运行时间 $b_{i,j}^{\min}$ 分别为 7, 5, 6, 5, 7 min. 根据京津高速线实际路情, 设置最小停站时间 $c_{i,j}^{\min}$ 为 2 min, 最小到达间隔时间 τ_{dd} 和最小发车间隔时间 τ_{ff} 均为 5 min.

按照假设 3 的条件, 下行线上除终点站天津站以外, 仅始发站北京南站附近存在北京动车段, 故令车站封锁发生于北京南站, 由此受车站封锁影响的列车可存放于北京南站及其附近的北京动车段. 需要说明的是, 车站封锁并非只能设置在始发站, 某些满足假设 3 条件的中间站 (例如, 沈阳北站、成都东站和武汉站等) 同样可设置本文所考虑的车站封锁.

铁路部门暂时没有对车站封锁等级进行确定划分, 为验证算法在不同车站封锁时长下的有效性, 且不对问题研究产生影响的前提下, 本文按照车站封锁时长大小设置 3 种不同的车站封锁等级, 具体如下. 车站封锁时长小于 60 min 为 I 级封锁, 车站封锁时长不小于 60 min 但小于 120 min 为 II 级封锁, 车站封锁时长不小于 120 min 为 III 级封锁. 其次, 为验证算法在不同运营时段的有效性, 统计京津高速线北京南站每小时发出的列车数量. 根据该数量的最大值和最小值, 将分别在 6∼9 点, 11∼14 点, 17∼19 点的运营时段划分为早高峰、午平峰、晚高峰. 综合上述不同的车站封锁等级和运营时段, 设置 9 个典型场景, 如表 1 所示. 具体信息包括场景序号及类型, 车站封锁时间范围 $[t_{\text{start}}, t_{\text{end}})$, 车站封锁时长 t_{blg}, 封锁区域内列车数量 N_{blg}, 以及各场景所考虑的列车数量 N_{con}.

注1 由于各典型场景下基本运行图中的所有列车并非都受到影响, 针对各场景只需要考虑部分的受影响列车. 基本做法是, 实际仿真实验前, 先通过多次交叉验证确定每个场景所考虑的列车数量 N_{con}.

表 1 车站封锁情况下的典型场景

Table 1 Typical scenarios under the station blockage

No.	Type	$[t_{\text{start}}, t_{\text{end}})$	t_{blg} (min)	N_{blg}	N_{con}
1	Morning peak, level I blockage	[6:00, 6:20)	20	4	5
2	Morning peak, level II blockage	[6:00, 7:20)	80	10	27
3	Morning peak, level III blockage	[6:00, 8:20)	140	17	43
4	Noon off-peak, level I blockage	[11:00, 11:30)	30	3	6
5	Noon off-peak, level II blockage	[11:00, 12:30)	90	9	18
6	Noon off-peak, level III blockage	[11:00, 13:30)	150	16	38
7	Evening peak, level I blockage	[17:00, 17:40)	40	6	16
8	Evening peak, level II blockage	[17:00, 18:20)	80	10	24
9	Evening peak, level III blockage	[17:00, 19:00)	120	16	34

5.2 进化算法及其运行参数

由于列车运行调整问题存在 NP 难特性和复杂时空约束, 难以针对问题设计特定的进化算法. 故选取以下典型的基本进化算法, 并在不同的种群染色体编码方法和种群初始化方法下进行计算实验.

(1) 差分进化 (differential evolution, DE) [24]: 本文首先选取 DE 作为进化计算的测试算法, 包含以下两点原因. 其一, DE 针对车间作业调度模型中的 NP 难特性具有较好的求解效果, 而很多研究学者通常针对列车运行调整问题建立车间作业调度模型 [7], 故 DE 可能比较适用于求解列车运行调整问题. 其二, 在求解调度优化问题时, 相较于粒子群算法及其变体, DE 有更好的求解效率和优化方案 [25].

(2) 加强精英保留的遗传算法 (strengthened elitist genetic algorithm, SEGA): 遗传算法是进化计算领域中比较常用且有效的算法, 为提升求解性能及效率, SEGA 将父代种群与交叉变异后的种群进行合并, 并从合并的个体中选出精英个体, 由此提高精英个体在进化计算过程中被选择的概率 [26].

上述进化算法代码已由 Jazzbin [27] 基于 Python 语言在 Geatpy 的进化算法工具箱中编写完成, Geatpy 采用 Numpy+mkl 和 C 内核, 拥有比 Java 和 Matlab 更快的求解速度. 本文根据 Geatpy 的问题接口编写列车运行调整模型, 在 Geatpy 的算法接口应用本文所提出的基于问题知识的运行图实时调整方法. 由于 Geatpy 中 DE 无排列编码方法, 本文采用随机键值 (random key) 方法 [28], 将实整数编码下的决策变量按数值大小关系进行排列, 由此将 DE 实整数编码下原有的连续实整数空间转化为排列空间, 成功地将本文所提出的排列编码方法应用于 DE 中.

仿真实验中的比较方法包括以下 3 种. (1) 商业优化软件 CPLEX: CPLEX 一旦求解完成列车运行调整模型, 可直接生成问题最优解, 故 CPLEX 用于验证最优改进进化算法所求的发车次序是否为最优解. (2) 先调度先服务 FSFS [7]: FSFS 生成基本运行图的发车次序, 与最优改进进化算法所求发车次序比较, 用于计算最优改进进化算法的提升效率. (3) 本文第 4.3 小节所提出的问题知识 PSK: 与最优改进进化算法所求发车次序进行比较, 用于验证 PSK 为初始种群所提供较优个体解的质量.

为了确定进化算法的运行参数, 针对表 1 所示的车站封锁影响下各典型场景, 在种群规模 N_p 为 40, 45, 50, 55, 60, 变异概率 p_m 为 0.5, 0.6, 0.7, 0.8, 0.9 的参数设置下均运行 25 次计算实验. 对不同参数下的实验结果进行灵敏度分析[1)], 确定种群规模 N_p 为 50, 变异概率 p_m 为 0.8. 在兼顾列车运行调整方案最优性和实时性两个指标的基础上, 设置排列编码和实整数编码下适应度函数的最大评价

1) https://github.com/RongshengWang/Problem-specific-knowledge-driven-real-time-rescheduling-approach-for-high-speed-train-operation/blob/main/5.2/5.2.1.rar.

次数 $\mathrm{NFE}_{\max}$ 分别为 400 和 10000. 根据 Geatpy 的默认参数设置 [27], DE 和 SEGA 算法下的交叉概率 p_c 分别为 0.5 和 0.7. 根据我国高铁实际路情, 定义列车运行调整实时性的时间标准为 10 min, 即算法在 10 min 内给出调整方案则满足实时性要求.

5.3 结果分析

为便于阐述实验结果, 定义以下缩写方式. 排列编码和实整数编码分别用 PM (permutation) 和 RI (real-integer) 表示, 随机种群初始化表示为 RP (random population), 基于问题知识的启发式种群初始化表示为 HP (hybrid population). 综合以上缩写方式表示不同编码方法和种群初始化方法组合下的基本进化算法和改进进化算法. 例如, PM-DE-RI 表示排列编码 (PM) 方法下采用随机种群初始化 (RP) 的改进 DE 算法. 为了客观比较 2 种编码方法的优劣, 并说明基于问题知识的启发式种群初始化下算法的性能提升, 仿真实验针对表 1 所示的每个场景, 在每个基本进化算法和改进进化算法上均运行 25 次, 并进行显著性水平为 0.05 的 Wilcoxon 秩和检验. 在上述所有算法的对比实验中, 令参数 S 为 Wilcoxon 秩和检验的评分. 当求解某个场景的目标函数值时, 若当前算法比本文其他某种算法更优, 则本次比较实验中 $S = 1$; 若前者性能比后者性能更差, 则本次比较实验中 $S = -1$; 若二者性能相当, 则本次比较实验中 $S = 0$.

5.3.1 不同编码方法下算法的性能比较

为验证本文所提出排列编码方法在减小列车总晚点时间上的有效性和求解效率, 选取表 1 中 3 个 I 级封锁下的场景 1, 4 和 7. 针对第 4.1 小节所述传统的接发车时刻实整数编码 (RI) 和本文所提出的发车次序排列编码 (PM), 均采用随机种群初始化 (RP) 方法, 并进行 25 次计算实验. 根据上述缩写方式说明, 接发车时刻实整数编码下的 2 种基本进化算法为 RI-DE-RP 和 RI-SEGA-RP. 应用发车次序排列编码的 2 种改进进化算法为 PM-DE-RP 和 PM-SEGA-RP. 比较上述 4 个算法的目标函数值 (Val, 均值 ± 标准差) 和求解时间 (Time, 均值 ± 标准差), 如表 2 所示. 其中, 加粗字体表示目标函数值 Val 最小和评分 S 最大的结果. 由于所有场景下当前算法需要与本文其他 3 种算法比较, Wilcoxon 秩和检验下评分 S 的最大值为 27, 最小值为 −27. 尽管仅计算 3 个 I 级封锁下的场景, 但足以说明排列编码优势.

根据表 2 所示结果, 有如下发现. (1) 应用排列编码的 PM-DE-RP 和 PM-SEGA-RP 针对表 1 中的 9 个场景均取得近似最优解, 并且 2 种改进进化算法在场景 1 (Val = 1080.0 ± 0.0) 和场景 4 (Val = 780.0 ± 0.0) 中获得方差为 0 的最优解 (已由 CPLEX 验证). (2) 采用接发车实整数编码的 RI-DE-RP 仅在场景 1 (Val = 1120.1 ± 196.5) 和场景 4 (Val = 817.9 ± 142.6) 中取得可行解, 但求解结果明显差于使用排列编码的 PM-DE-RP 在场景 1 (Val = 1080.0 ± 0.0) 和场景 4 (Val = 780.0 ± 0.0) 中的平均目标函数值. 其他应用实整数编码的基本进化算法在其他场景中都未取得可行解 (表 2 中 NA). 其原因是, 在适应度函数最大评价次数足够大的前提下, 实整数编码搜索效率较低, 实时性较差, 难以获取可行解. (3) 对于评分值 S, PM-DE-RP ($S = 20$) 大于 PM-SEGA-RP ($S = 18$), 并且前者在 9 个场景中均取得平均目标函数最小的求解方案 (表 2 中加粗的 Val). 由此说明在应用排列编码但未采用启发式种群初始化的前提下, DE 改进进化算法 (PM-DE-RP) 的性能优于 SEGA 改进进化算法 (PM-SEGA-RP). (4) 应用排列编码的 2 种改进进化算法 (PM-DE-RP, PM-SEGA-RP) 的平均求解时间 (Time) 都小于 18 s, 满足实时性的求解要求, 且远小于采用实整数编码的 2 种基本进化算法 (RI-DE-RP, RI-SEGA-RP) 的求解时间 (400∼1378 s). 通过仿真实验的定量结果验证, 基于发车次序的排列编码方法减小种群个体向量维度的同时, 有效减少了问题求解过程中解空间的无效搜索, 提升了进化算法的求解质量和

表 2 RI-DE-RP, PM-DE-RP, RI-SEGA-RP, PM-SEGA-RP 求解 3 个 I 级封锁下场景的实验结果

Table 2 Experiment results of RI-DE-RP, PM-DE-RP, RI-SEGA-RP, and PM-SEGA-RP in the three scenarios of blockage level I, where NA means that no feasible solutions are found in the current algorithm

No.	RI-DE-RP		PM-DE-RP		RI-SEGA-RP		PM-SEGA-RP	
	Val (min)	Time (s)	Val (min)	Time (s)	Val (min)	Time (s)	Val (min)	Time (s)
1	1120.1 ± 196.5	400.10 ± 5.18	**1080.0 ± 0.0**	1.95 ± 0.05	NA	418.22 ± 8.38	**1080.0 ± 0.0**	2.10 ± 0.04
2	NA	954.22± 5.46	**13211.4 ± 22.4**	10.41 ± 0.10	NA	986.98± 18.29	13264.9 ± 83.0	10.47 ± 0.22
3	NA	1349.04 ± 52.62	**39254.3 ± 60.5**	17.25 ± 0.16	NA	1377.51 ± 9.76	39291.3 ± 140.8	16.87 ± 0.27
4	817.9 ± 142.6	430.45 ± 3.57	**780.0 ± 0.0**	2.22± 0.03	NA	449.66 ± 3.40	**780.0 ± 0.0**	2.35 ± 0.03
5	NA	726.93 ± 4.70	**9340.0 ± 0.0**	6.45 ± 0.07	NA	746.40 ± 4.73	9347.5 ± 29.7	6.54 ± 0.15
6	NA	1214.81 ± 6.27	**30650.7 ± 55.5**	14.70 ± 0.42	NA	1254.74 ± 6.24	30693.1 ± 151.3	14.29 ± 0.48
7	NA	680.30 ± 6.21	**3764.0 ± 0.0**	5.87 ± 0.04	NA	711.04 ± 25.44	3767.7 ± 14.5	6.05 ± 0.10
8	NA	972.29 ± 3.39	**13094.0 ± 0.0**	9.85 ± 0.09	NA	997.64 ± 7.07	13137.4 ± 74.5	10.22 ± 0.11
9	NA	1143.04 ± 8.32	**27887.4 ± 37.9**	12.75 ± 0.24	NA	1166.45 ± 21.87	27925.4 ± 80.5	12.54 ± 0.28
S	−16		**20**		−20		18	

效率.

5.3.2 启发式种群初始化下算法的性能提升

为验证本文所提出基于问题知识的启发式种群初始化下进化算法的性能提升, 以及说明不同进化算法对启发式初始化种群的敏感性问题. 选取表 1 中的所有场景, 针对第 5.3.1 小节 2 种改进进化算法 (PM-DE-RP, PM-SEGA-RP) 应用启发式种群初始化的改进, 生成另外 2 种改进进化算法: PM-DE-HP 和 PM-SEGA-HP. 计算上述 4 个改进进化算法在 25 次计算实验后的目标函数值 (Val, 均值 ± 标准差) 和求解时间 (Time, 均值 ± 标准差), 如表 3 所示. 加粗字体表示目标函数值 Val 最小和评分 S 最大的结果.

根据表 3 所示结果, 有如下发现. (1) 在基于问题知识的启发式种群初始化 (HP) 改进下, PM-SEGA-HP 消除原有 PM-SEGA-RP 在场景 5 (Val = 9347.5 ± 29.7), 场景 7 (3767.7 ± 14.5) 和场景 8 (13137.4 ± 74.5) 中目标函数值的标准差 (Val = 9340.0 ± 0.0, Val = 3764.0 ± 0.0, Val = 13094.0 ± 0.0). (2) 由评分结果 S 可知, 启发式种群初始化 (HP) 改进下, DE 的 S 值由 −6 提升至 4, SEGA 的 S 值由 −10 提升至 12, 由此可见, SEGA 的提升效果更大. 且改进后的 PM-SEGA-HP ($S = 12$) 优于改进后的 PM-DE-HP ($S = 4$), 是提升效果最好且评分值最高的改进进化算法. (3) 由表 3 中求解时间 (Time) 结果可知, 4 个改进进化算法针对所有场景的求解时间都满足列车运行调整 10 min 的实时性要求; (4) 同样, 根据求解时间 (Time) 结果, 采用启发式种群初始化 (HP) 的改进后, DE 和 SEGA 的求解效率并未降低, 甚至在某些复杂的场景中获得一定提升. 例如场景 9 中, PM-SEGA-HP 的求解时间 (Time = 11.66 ± 0.12) 小于 PM-SEGA-RP (Time = 12.54 ± 0.28), PM-DE-HP 的求解时间 (Time = 12.31 ± 0.15) 小于 PM-DE-RP (Time = 12.75 ± 0.24).

由于 4 个改进进化算法在求解 3 个 III 级封锁下场景 (场景 3, 6, 9) 的目标函数值具有较大差异, 故绘制上述 3 个场景下目标函数值的箱型图, 如图 3 所示. 由图 3 可知, 25 次计算实验中目标函数值的异常值、中位数、四分位数等数据信息. 同时, 可以发现, 相较于本文其他 3 种算法 (PM-DE-RP, PM-SEGA-RP, PM-DE-HP), PM-SEGA-HP 在 3 个 III 级封锁场景中对应的目标函数虽然存在少许异常值, 但求解结果数据分布相对最集中, 算法稳定性最好. 计算上述 3 个场景下, 4 个改进进化算法

表 3　PM-DE-RP, PM-SEGA-RP, PM-DE-HP, PM-SEGA-HP 求解 9 个场景的实验结果

Table 3　Experiment results of PM-DE-RP, PM-SEGA-RP, PM-DE-HP, and PM-SEGA-HP in the nine scenarios

No.	PM-DE-RP		PM-SEGA-RP		PM-DE-HP		PM-SEGA-HP	
	Val (min)	Time (s)	Val (min)	Time (s)	Val (min)	Time (s)	Val (min)	Time (s)
1	**1080.0 ± 0.0**	1.95 ± 0.05	**1080.0 ± 0.0**	2.10 ± 0.04	**1080.0 ± 0.0**	1.95 ± 0.06	**1080.0 ± 0.0**	2.08 ± 0.06
2	13211.4 ± 22.4	10.41 ± 0.10	13264.9 ± 83.0	10.47 ± 0.22	**13194.4 ± 8.6**	10.32 ± 0.11	13211.3 ± 56.1	9.94 ± 0.20
3	39254.3 ± 60.5	17.25 ± 0.16	39291.3 ± 140.8	16.87 ± 0.27	39181.9 ± 30.6	17.03 ± 0.36	**39122.7 ± 14.2**	15.96 ± 0.31
4	**780.0 ± 0.0**	2.22 ± 0.03	**780.0 ± 0.0**	2.35 ± 0.03	**780.0 ± 0.0**	2.26 ± 0.03	**780.0 ± 0.0**	2.38 ± 0.05
5	**9340.0 ± 0.0**	6.45 ± 0.07	9347.5 ± 29.7	6.54 ± 0.15	**9340.0 ± 0.0**	6.48 ± 0.07	**9340.0 ± 0.0**	6.45 ± 0.06
6	30650.7 ± 55.5	14.70 ± 0.42	30693.1 ± 151.3	14.29 ± 0.48	30599.4 ± 39.3	14.21 ± 0.52	**30552.6 ± 3.1**	13.20 ± 0.46
7	**3764.0 ± 0.0**	5.87 ± 0.04	3767.7 ± 14.5	6.05 ± 0.10	**3764.0 ± 0.0**	5.80± 0.03	**3764.0 ± 0.0**	5.90 ± 0.04
8	**13094.0 ± 0.0**	9.85 ± 0.09	13137.4 ± 74.5	10.22 ± 0.11	**13094.0 ± 0.0**	9.66 ± 0.05	**13094.0 ± 0.0**	9.84 ± 0.06
9	27887.4 ± 37.9	12.75 ± 0.24	27925.4 ± 80.5	12.54 ± 0.28	27853.9 ± 34.0	12.31 ± 0.15	**27831.8 ± 33.9**	11.66 ± 0.12
S	−6		−10		4		**12**	

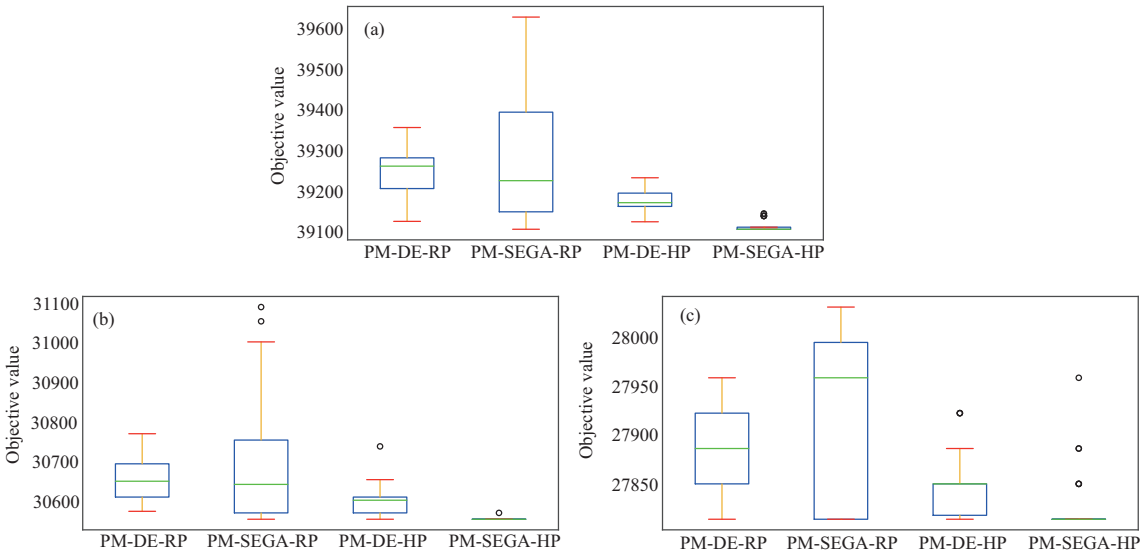

图 3　(网络版彩图) 3 个 III 级封锁下的场景中 4 个改进进化算法目标函数值的箱型图

Figure 3　(Color online) Box plots of objective values of the four improved evolutionary algorithms in the three scenarios of blockage level III. (a) Scenario 3; (b) scenario 6; (c) scenario 9

在 25 次计算实验中每一代函数评价下的平均目标函数值, 绘制其收敛曲线, 如图 4 所示. 由图 4 可知, 相较于使用随机种群初始化 (RP) 的 PM-DE-RP 和 PM-SEGA-RP, 采用基于问题知识的启发式种群初始化 (HP) 的 PM-DE-HP 和 PE-SEGA-HP 都能更快地收敛于各自近似最优的调整方案. 并且, PM-SEGA-HP 在 4 个改进进化算法中具有最快的算法前期收敛速度和最小的平均目标函数值.

5.3.3　各调整方法性能对比

为选出最优的改进进化算法, 比较 PM-DE-HP 和 PM-SEGA-HP 在算法提升效果 (5.3.2 小节的结论 (2)), 评分值 (表 3 中 S 值), 取得最小平均目标函数值的个数 (表 3 标粗的数据个数) 和算法前期

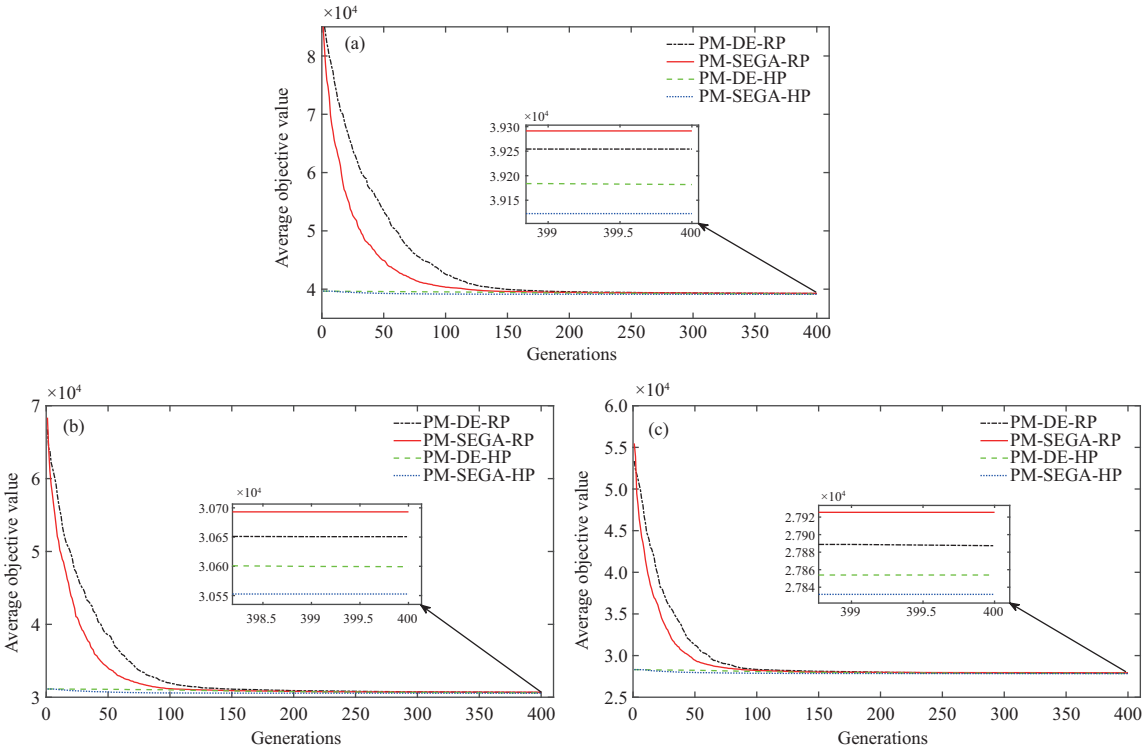

图 4 (网络版彩图) 3 个 III 级封锁下的场景中 4 个改进进化算法平均目标函数值的收敛曲线

Figure 4 (Color online) Convergence curves of average objective values of the four improved evolutionary algorithms in the three scenarios of blockage level III. (a) Scenario 3; (b) scenario 6; (c) scenario 9

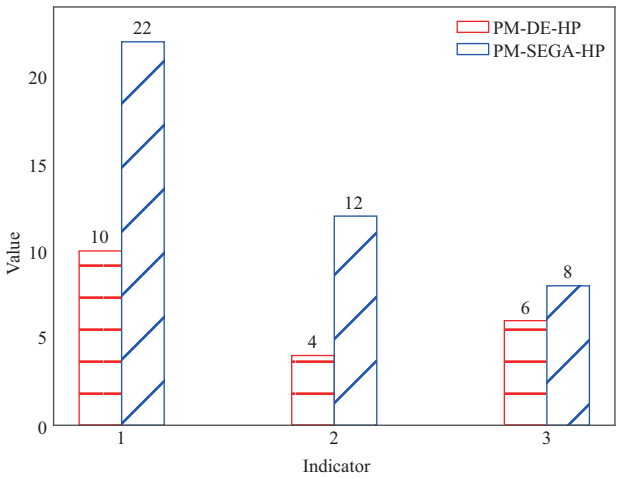

图 5 (网络版彩图) PM-DE-HP 与 PM-SEGA-HP 的指标对比结果

Figure 5 (Color online) Indicators comparison results of PM-DE-HP and PM-SEGA-HP. Indicator 1: the degree of performance improvement; Indicator 2: the value of S; Indicator 3: the number of acquiring the minimum average objective value

收敛速度 (图 4 蓝色点线) 4 个指标上的结果. 前 3 个指标的对比结果如图 5 所示, PM-SEGA-HP 在上述 3 个指标上均优于 PM-DE-HP. 第 4 个指标算法已由 5.3.2 小节分析可知, PM-SEGA-HP 具有最

表 4 PM-SEGA-HP, CPLEX, FSFS, PSK 求解 9 个场景的优化方案、目标函数值和求解时间

Table 4 Optimized solutions, objective values (Value) and computation times (Time) of PM-SEGA-HP, CPLEX, FSFS, and PSK in the nine scenarios, where NA means that CPLEX cannot provide optimal solutions within 10 min

No.	Method	Optimized solution	Value (min)	Time (s)
1	PM-SEGA-HP	0, 1, 2, 3, 4	**1080**	2.08 ± 0.06
	CPLEX	0, 1, 2, 3, 4	**1080**	0.73
	FSFS	0, 1, 2, 3, 4	**1080**	<0.01
	PSK	0, 1, 2, 3, 4	**1080**	<0.01
2	PM-SEGA-HP	0, 1, 2, 6, 12, 5, 7, 3, 10, 11, 8, 14, 16, 17, 18, 19, 15, 20, 21, 22, 4, 13, 24, 23, 9, 25, 26	13188	9.94 ± 0.20
	CPLEX	NA	NA	NA
	FSFS	0, 1, . . . , 25, 26	13773	<0.01
	PSK	0, 1, 2, 3, 4, 5, 6, 7, 8, 4, 9, 10, . . . , 25, 26	13629	<0.01
3	PM-SEGA-HP	2, 14, 6, 0, 19, 1, 7, 10, 21, 17, 11, 18, 8, 3, 5, 16, 12, 26, 25, 23, 20, 28, 27, 15, 31, 32, 30, 33, 29, 22, 24, 4, 37, 36, 13, 38, 34, 39, 40, 9, 35, 41, 42	39114	15.96 ± 0.31
	CPLEX	NA	NA	NA
	FSFS	0, 1, . . . , 41, 42	40054	<0.01
	PSK	0, 1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 14, 16, 15, 4, 13, 9, 17, 18, . . . , 41, 42	39670	<0.01
4	PM-SEGA-HP	0, 1, 2, 3, 4, 5	**780**	2.38 ± 0.05
	CPLEX	0, 1, 2, 3, 4, 5	**780**	0.28
	FSFS	0, 1, 2, 3, 4, 5	**780**	<0.01
	PSK	0, 1, 2, 3, 4, 5	**780**	<0.01
5	PM-SEGA-HP	4, 2, 8, 9, 0, 6, 3, 10, 12, 1, 13, 14, 15, 7, 5, 11, 16, 17	9340	6.45 ± 0.06
	CPLEX	NA	NA	NA
	FSFS	0, 1, . . . , 16, 17	9664	<0.01
	PSK	0, 1, 2, 3, 4, 6, 8, 7, 5, 9, 10, . . . , 16, 17	9580	<0.01
6	PM-SEGA-HP	2, 4, 13, 8, 0, 1, 10, 3, 12, 6, 9, 21, 20, 15, 14, 22, 23, 24, 25, 16, 18, 19, 26, 29, 30, 31, 27, 28, 32, 7, 33, 34, 35, 11, 5, 17, 36, 37	30552	13.20 ± 0.46
	CPLEX	NA	NA	NA
	FSFS	0, 1, . . . , 36, 37	31456	<0.01
	PSK	0, 1, 2, 3, 4, 6, 8, 9, 10, 12, 13, 14, 15, 7, 5, 11, 16, 17, . . . , 36, 37	31132	<0.01
7	PM-SEGA-HP	0, 2, 1, 6, 4, 7, 3, 5, 9, 10, 11, 12, 13, 8, 14, 15	3764	5.90 ± 0.04
	CPLEX	NA	NA	NA
	FSFS	0, 1, . . . , 14, 15	3856	<0.01
	PSK	0, 1, . . . , 14, 15	3856	<0.01
8	PM-SEGA-HP	0, 7, 9, 10, 12, 3, 2, 11, 4, 15, 14, 1, 6, 16, 17, 18, 5, 13, 19, 20, 21, 8, 22, 23, 24, 25, 26, 27	13094	9.84 ± 0.06
	CPLEX	NA	NA	NA
	FSFS	0, 1, . . . , 26, 27	13506	<0.01
	PSK	0, 1, 2, 3, 4, 6, 7, 9, 5, 8, 10, 11, . . . , 26, 27	13394	<0.01
9	PM-SEGA-HP	11, 2, 9, 14, 0, 15, 10, 1, 3, 4, 6, 12, 20, 16, 7, 17, 19, 24, 22, 21, 25, 26, 27, 28, 29, 13, 30, 18, 23, 5, 31, 32, 8, 33, 34	27816	11.66 ± 0.12
	CPLEX	NA	NA	NA
	FSFS	0, 1, . . . , 33, 34	28632	<0.01
	PSK	0, 1, 2, 3, 4, 6, 7, 9, 10, 11, 12, 14, 15, 5, 13, 8, 16, 17, . . . , 33, 34	28316	<0.01

表 5 求解 9 个场景的 RPD 结果 (%)

Table 5 RPD results of the nine scenarios

Algorithm	No. 1	No. 2	No. 3	No. 4	No. 5	No. 6	No. 7	No. 8	No. 9
FSFS	0.00	4.44	2.40	0.00	3.47	2.96	2.44	3.15	2.93
PSK	0.00	3.34	1.42	0.00	2.57	1.90	2.44	2.29	1.80

快的算法前期收敛速度. 故选择 PM-SEGA-HP 作为本文的最优改进进化算法, 并分别在 9 个场景各自的 25 次计算实验中, 选择 PM-SEGA-HP 使列车总晚点时间最小的发车次序, 并与 CPLEX, FSFS, PSK 方法下的发车次序对比. 上述 4 种调整方法所求发车次序下的目标函数值和求解时间如表 4 所示, 有如下发现.

(1) 对于 2 个 I 级封锁下的场景 1 和 4, PM-SEGA-HP 可获取与 CPLEX 相同的最优发车次序 (表 4 中加粗字体). 但在其他 7 个场景中, CPLEX 都无法在 10 min 内获得最优解 (表 4 中的 NA), 而 PE-SEGA-HP 在 16 s 的平均求解时间内均可给出近似最优解.

(2) PM-SEGA-HP 方法下的目标函数值对应不止一个发车次序调整方案[2)], 说明表 1 中的某些场景存在多模态性质. 进一步, 运行图中同质性的列车具有相同的运行等级, 导致列车运行调整问题本质是个多模态问题. 随着封锁时长增大, CPLEX 无法在 10 min 内取得最优解. 其次, 多个发车次序调整方案虽然针对对应场景都是可行的, 但某些调整方案下列车的调整范围较大, 不利于调度员向司机和车站下达调度命令. 因此, 可根据调度员具体需求, 剔除某些列车调整范围比较大的求解方案.

(3) 比较 FSFS, PSK 与 PE-SEGA-HP 的目标函数值, 说明最优改进进化算法 PE-SEGA-HP 在减小列车总晚点时间上的性能提升, 以及验证 PSK 为初始种群提供较优个体解的质量. 定义相对百分比偏差 (relative percentage deviation, RPD) 定量说明上述指标, 即

$$\mathrm{RPD} = (\mathrm{alg} - \mathrm{bst})/\mathrm{bst}, \tag{9}$$

其中, alg 表示 FSFS 或者 PSK 方法求解某场景的目标函数值, bst 表示最优改进进化算法 PE-SEGA-HP 求解对应场景的目标函数值. 表 5 列出了 9 个场景下 RPD 的计算结果. 当算法为 FSFS 时, RPD 结果表示相较于基本运行图的图定发车次序, PM-SEGA-HP 所给出发车次序下求解质量的提升大小. 当算法为 PSK 时, 表明问题知识可为初始种群提供一个与 PE-SEGA-HP 所求方案误差在 4% 以内的种群个体.

最后, 同样选取 3 个 III 级封锁下的场景, 给出 PM-SEGA-HP 发车次序下的实绩运行图, 如图 6 所示, 蓝色实线标注车站封锁的时间范围. 由图 6 可知, 当列车实际停站次数越少或实际区间运行时间越小时, 列车运行等级越高, 在封锁车站越早发车.

6 结论

本文针对车站封锁情况, 提出了发车次序的排列编码方法、启发式消除约束的解码方法, 以及启发式种群初始化, 选取进化算法 DE 和 SEGA, 从仿真实验验证上述改进在进化计算求解效率上的提升. 选取其中的最优改进进化算法 PE-SEGA-HP, 并与 CPLEX, FSFS 和 PSK 方法在求解方案上对比, 验证了本文所提出方法在减小列车总晚点时间的有效性和求解效率. 本文方法可作为算法模块为

2) https://github.com/RongshengWang/Problem-specific-knowledge-driven-real-time-rescheduling-approach-for-high-speed-train-operation/blob/main/5.2/5.3.3.csv.

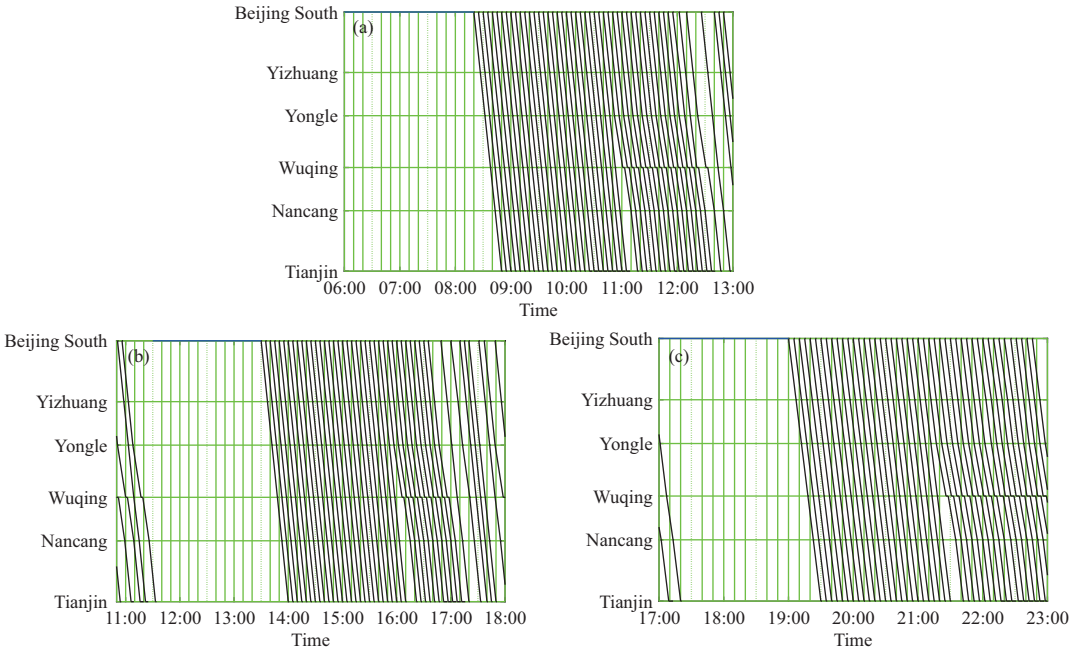

图 6 (网络版彩图) PM-SEGA-HP 求解 3 个 III 级封锁下场景的实绩运行图

Figure 6 (Color online) Rescheduled timetables of PM-SEGA-HP in the three scenarios of blockage level III. The solid blue line represents the time period under the station blockage. (a) Scenario 3; (b) scenario 6; (c) scenario 9

调度员实时提供运行图调整的建议方案. 未来考虑在模型中加入动车组运用计划、客流、车站进路等约束, 进一步提高调整方案的可行性和实用性.

参考文献

1 Lusby R M, Haahr J T, Larsen J, et al. A branch-and-price algorithm for railway rolling stock rescheduling. Transpation Res Part B-Meth, 2017, 99: 228–250

2 Cacchiani V, Huisman D, Kidd M, et al. An overview of recovery models and algorithms for real-time railway rescheduling. Transpation Res Part B-Meth, 2014, 63: 15–37

3 Pellegrini P, Pesenti R, Rodriguez J. Efficient train re-routing and rescheduling: valid inequalities and reformulation of RECIFE-MILP. Transpation Res Part B-Meth, 2019, 120: 33–48

4 Zhang J J, Hu W F, Peng T, et al. Moving block principle-based multi-strategy optimal scheduling method for trains in case of segment blockages. Sci Sin Inform, 2021, 51: 413–429 [张俊杰, 胡文峰, 彭涛, 等. 基于移动闭塞原理的区间中断下列车多策略优化调度方法. 中国科学: 信息科学, 2021, 51: 413–429]

5 Zhu Y Q, Goverde R M P. Railway timetable rescheduling with flexible stopping and flexible short-turning during disruptions. Transpation Res Part B-Meth, 2019, 123: 149–181

6 Corman F, Quaglietta E, Goverde R M P. Automated real-time railway traffic control: an experimental analysis of reliability, resilience and robustness. Transpation Plann Tech, 2018, 41: 421–447

7 Xu P J, Corman F, Peng Q Y, et al. A train rescheduling model integrating speed management during disruptions of high-speed traffic under a quasi-moving block system. Transpation Res Part B-Meth, 2017, 104: 638–666

8 Ning B. A number of scientific and technical problems in intelligent transportation. Sci Sin Inform, 2018, 48: 1264–1269 [宁滨. 智能交通中的若干科学和技术问题. 中国科学: 信息科学, 2018, 48: 1264–1269]

9 Šemrov D, Marsetič R, Žura M, et al. Reinforcement learning approach for train rescheduling on a single-track railway. Transpation Res Part B-Meth, 2016, 86: 250–267

10 Khadilkar H. A scalable reinforcement learning algorithm for scheduling railway lines. IEEE Trans Intell Transp Syst,

2019, 20: 727–736
11 Dai X W, Cheng L J, Cui D L, et al. Rescheduling of high-speed trains: a reinforcement learning approach. Sci Sin Inform, 2022, 52: 890–906 [代学武, 程丽娟, 崔东亮, 等. 基于强化学习的高速列车群运行调整方法. 中国科学: 信息科学, 2022, 52: 890–906]
12 Wang R S, Zhou M, Li Y D, et al. A timetable rescheduling approach for railway based on Monte Carlo tree search. In: Proceedings of the 22nd IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, 2019. 3738–3743
13 Ning L B, Li Y D, Zhou M, et al. A deep reinforcement learning approach to high-speed train timetable rescheduling under disturbances. In: Proceedings of the 22nd IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, 2019. 3769–3774
14 Eiben A E, Smith J E. Introduction to Evolutionary Computing. 2nd ed. Berlin: Springer Press, 2015. 13–18
15 Eaton J, Yang S, Gongora M. Ant colony optimization for simulated dynamic multi-objective railway junction rescheduling. IEEE Trans Intell Transp Syst, 2017, 18: 2980–2992
16 Tang L H, D'Ariano A, Xu X F, et al. Scheduling local and express trains in suburban rail transit lines: mixed-integer nonlinear programming and adaptive genetic algorithm. Comput Operations Res, 2021, 135: 105436
17 Lin B, Yu S P, Liu Z Y, et al. High-speed train dynamic scheduling method based on improved particle swarm optimization algorithm. China Control Eng, 2021, 28: 1334–1341 [林博, 俞胜平, 刘子源, 等. 基于改进粒子群算法的高铁列车动态调度. 控制工程, 2021, 28: 1334–1341]
18 Guo M, Xin B, Chen J, et al. Multi-agent coalition formation by an efficient genetic algorithm with heuristic initialization and repair strategy. Swarm Evolary Computation, 2020, 55: 100686
19 Wang Y P, Xin B, Chen J. An adaptive memetic algorithm for the joint allocation of heterogeneous stochastic resources. IEEE Trans Cybern, 2022, 52: 11526–11538
20 Zhang Q, Jin J, Liu Y, et al. Interim technical specification for intelligent centralized traffic control (CTC) system. China Railway, TJ/DW208-2019. 2019-9-27 [张琦, 靳俊, 刘岩, 等. 智能调度集中系统暂行技术条件. 中国国家铁路集团有限公司, TJ/DW208-2019. 2019 年 9 月 27 日]
21 Yang X, Li X, Gao Z Y, et al. A cooperative scheduling model for timetable optimization in subway systems. IEEE Trans Intell Transp Syst, 2013, 14: 438–447
22 Wang M M, Wang L, Xu X Y, et al. Genetic algorithm-based particle swarm optimization approach to reschedule high-speed railway timetables: a case study in China. J Adv Transpation, 2019, 2019: 1–12
23 Wang J F, Wang J G, Roberts C, et al. Parallel monitoring for the next generation of train control systems. IEEE Trans Intell Transp Syst, 2015, 16: 330–338
24 Das S, Suganthan P N. Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Computat, 2011, 15: 4–31
25 Chen W, Panahi M, Pourghasemi H R. Performance evaluation of GIS-based new ensemble data mining techniques of adaptive neuro-fuzzy inference system (ANFIS) with genetic algorithm (GA), differential evolution (DE), and particle swarm optimization (PSO) for landslide spatial modelling. Catena, 2017, 157: 310–324
26 Zhang H Z, Liu F, Zhou Y Y, et al. A hybrid method integrating an elite genetic algorithm with tabu search for the quadratic assignment problem. Inf Sci, 2020, 539: 347–374
27 Jazzbin J. Geatpy: the genetic and evolutionary algorithm toolbox with high performance in Python. Version 2.6.0. 2020. http://geatpy.com
28 Baioletti M, Milani A, Santucci V. Algebraic particle swarm optimization for the permutations search space. In: Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastián, 2017. 1587–1594

Real-time rescheduling approach of train operation for high-speed railways using problem-specific knowledge under a station blockage

Rongsheng WANG[1,2,4], Qi ZHANG[2,4*], Tao ZHANG[2,4], Peng LIN[3,4], Shuxin DING[2,4] & Zhiming YUAN[2,4]

1. *Postgraduate Department, China Academy of Railway Sciences, Beijing* 100081*, China*;
2. *Signal and Communication Research Institute, China Academy of Railway Sciences Corporation Limited, Beijing* 100081*, China*;
3. *School of Automation, Central South University, Changsha* 410083*, China*;
4. *The Center of National Railway Intelligent Transportation System Engineering and Technology, China Academy of Railway Sciences Corporation Limited, Beijing* 100081*, China*
* Corresponding author. E-mail: zhangqi@rails.cn

Abstract This study considers the train timetable as the problem object to investigate the station blockage caused by emergencies. Based on the evolutionary algorithm framework, a real-time rescheduling approach using problem-specific knowledge is proposed. This approach ensures the safety and efficiency of high-speed rail operations and the satisfaction and comfort of passengers by minimizing the total train delay. First, a permutation encoding method is developed to reduce invalid searches in the search space based on the rescheduling strategy of reordering train departure sequences. Next, according to the train operation tracking mode, called tracking interval control, a heuristic decoding method is designed to eliminate all constraints of train operation to improve efficiency. Finally, the dispatcher's experience in adjusting the train timetable is used as the problem-specific knowledge to initialize the initial population of evolutionary computing. A heuristic population initialization method based on problem-specific knowledge is employed to speed up the algorithm convergence in the early stage and improve solution equality. Furthermore, the Beijing–Tianjin high-speed railway line is used as an example. Nine typical scenarios with different blockage durations of 20–150 min under the station blockage are installed at the Beijing South station. The strengthened elitist genetic algorithm (SEGA) and differential evolution (DE) are selected to perform the simulation using different combinations of the real-integer or permutation encoding and random or heuristic population initialization, respectively. The simulation results indicate that, compared with the real-integer encoding that cannot obtain feasible solutions, the two evolutionary algorithms can provide the rescheduling solution with a smaller total train delay in an average time of 9 s after using the permutation encoding. Besides, the results of the two evolutionary algorithms improved by the heuristic population initialization can quickly converge to a quasi-optimal solution. Finally, the SEGA in the permutation encoding with the heuristic population initialization is chosen as the optimal improved evolutionary algorithm. This improved evolutionary algorithm can provide quasi-optimal solutions in 20 s in the seven scenarios where the CPLEX cannot provide optimal solutions in 10 min.

Keywords high-speed railway, train rescheduling, station blockage, evolutionary computing, genetic algorithm, permutation-based optimization