A metaheuristic algorithm for a locomotive routing problem arising in the steel industry

A Metaheuristic Algorithm for a Locomotive Routing Problem Arising in the Steel Industry

Baobin Huang[a,b], Lixin Tang[a,*], Roberto Baldacci[c], Gongshu Wang[d], Defeng Sun[e]

[a]*National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University, Shenyang 110819, China*
[b]*Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, China*
[c]*Division of Engineering Management and Decision Sciences, College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar*
[d]*Liaoning Engineering Laboratory of Data Analytics and Optimization for Smart Industry, Shenyang 110819, China*
[e]*Liaoning Key Laboratory of Manufacturing System and Logistics Optimization, Shenyang 110819, China*

Abstract

In this study, we address the locomotive routing problem faced in the steel industry. During steel production, locomotives are employed to move torpedo cars, which transport molten iron, between blast furnaces and converters. The resulting complex pickup and delivery routing system which features multiple types of practical constraints, such as hard time windows, incompatibility of requests, variable locomotives capacity, and last-in-first-out constraints, constitutes this problem. Herein, three-index and modified group-based formulations are described and an adaptive large neighborhood search (ALNS) algorithm is proposed as a solution. The ALNS algorithm relies on both adapted and novel removal and insertion operators and a feature-based local search procedure. The mathematical formulations and the ALNS algorithm are evaluated on instances generated according to the actual production process. The results validate the effectiveness of the algorithm. A comparison with the manual plan also verifies the quality of the solutions produced by the algorithm.

Keywords: routing, steel industry, last-in-first-out constraints, local search, adaptive large neighborhood search.

1. Introduction

The steel industry is characterized by large-scale production, high energy consumption, and high pollution. The transformation from iron ore to end products is characterized by several complex problems, such as routing at mining areas (Moradi Afrapoli & Askari-Nasab, 2019), storage space allocation (Sun et al., 2020a), and continuous casting and hot rolling (Tang et al., 2001, 2014; Tang & Meng, 2021). Here, the problem was studied in the context of molten iron allocation and transportation. Among the stages involved in the production of steel, iron-making and steel-making are two production processes consuming the largest amounts of energy. Molten iron is transported from the iron-making facilities to the steel-making facilities by a special type of thermally insulated container, called torpedo car (TPC). Therefore, with regard to the

*Corresponding author
Email addresses: huangbaobinyc@163.com (Baobin Huang), lixintang@mail.neu.edu.cn (Lixin Tang), rbaldacci@hbku.edu.qa (Roberto Baldacci), wanggongshu@ise.neu.edu.cn (Gongshu Wang), sundefeng@ise.neu.edu.cn (Defeng Sun)

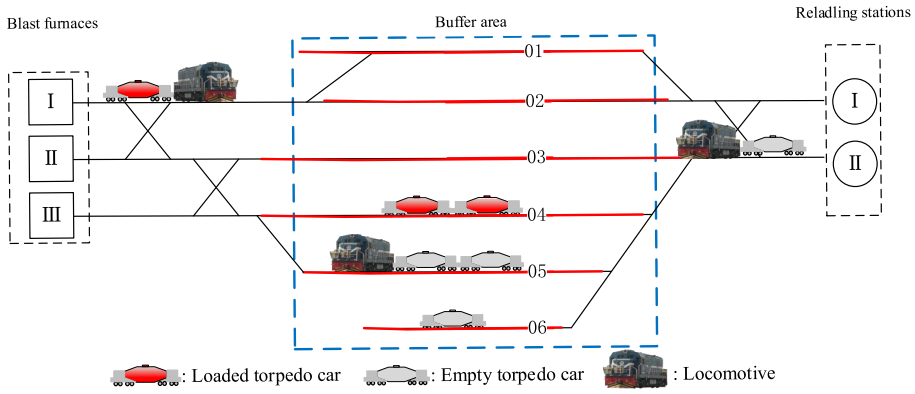

Fig. 1. Layout of production facilities and railway transportation network.

steel industry, there exist a demand for efficient molten iron transportation scheduling solutions, in order to guarantee continuous production and reduce the temperature loss related to energy consumption.

Fig.1 depicts the layout of the production facilities and the railway transportation network of the steel company considered in our study. The reladling stations (RSs), part of the steel-making shops for unloading the molten iron, and blast furnaces (BFs) are situated at different locations along the in-plant railroad network. Between the BFs and RSs, there is a buffer area (BA), comprising several long rails (the red lines 01-06 in Fig.1) used for parking locomotives and TPCs. A TPC is not equipped with an engine and is instead moved by a locomotive. Owing to the limited carrying capacity of locomotives, at most, two loaded or four empty TPCs can be towed by one locomotive at a time. Moreover, according to operation rules, loaded and empty TPCs cannot be towed by the same locomotive simultaneously. As shown in Fig.2, empty TPCs are loaded with molten iron at BFs (referred to as loaded TPCs) and then picked up by locomotives and delivered to the RSs. After unloading at the RSs, the empty TPCs are picked up by locomotives and delivered back to the BA and then to the BFs, where they wait for the next loading operation. There are four types of *transportation requests* as follows:

(i) BF→BA: transportation of a loaded TPC from BF to BA.
(ii) BA→RS: transportation of a loaded TPC from BA to RS.
(iii) RS→BA: transportation of an empty TPC from RS to BA.
(iv) BA→BF: transportation of an empty TPC from BA to BF.

Based on the production schedule, restrictions pertaining to the pickup and delivery time (modeled through suitable time windows, see Section 3) are imposed on each transportation request. Imposing time restrictions on the pickup and delivery of transportation requests ensures a continuous process and prevents congestion at the BFs and RSs.

If a locomotive collects more than one TPC, the pickup operations are termed as *coupling* operations, and the corresponding *decoupling* delivery operations must follow the last-in-first-out (LIFO) rule. Moreover, a pickup operation is forbidden for a locomotive once the decoupling delivery of any TPC has commenced, unless all the coupling TPCs have been delivered. These restrictions aim to reduce the complexity of the transportation operations and, in particular, to reduce the risk of accidents through prevention.

A critical factor in molten iron transportation is the temperature drop. If the temperature falls below the standard temperature required by the production process at RSs, an expensive reheating operation will be necessary. Therefore, the loaded TPCs must be delivered to RSs within a predefined time window to ensure

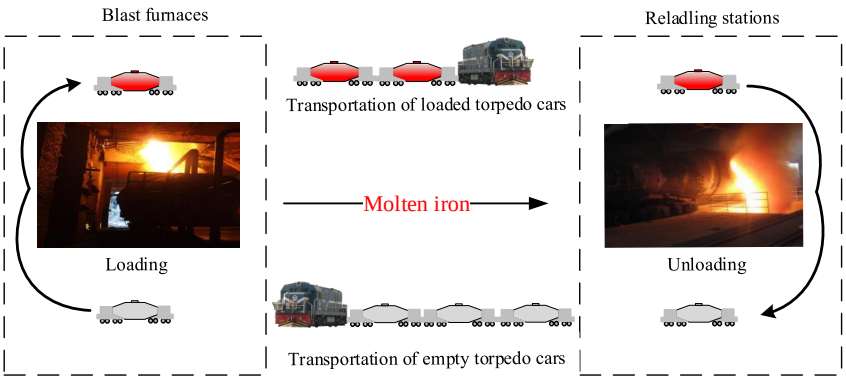

Fig. 2. Transportation process of molten iron.

that the temperature of molten iron is above the standard temperature. In practice, the temperature of the molten iron in a TPC reduces by approximately one degree Celsius per minute. Therefore, this temperature variation (BA→RS) can be represented by the waiting time, which refers to the period of time from loading completion to the arrival at the RS. Moreover, fuel consumption is related to the travel time of locomotives.

The problem addressed in this study is a variant of the pickup and delivery problem with time windows (PDPTW) in which a homogeneous fleet of vehicles (locomotives) is available, with different start and end locations for the vehicles, and a set of heterogeneous requests must be routed subject to time windows, capacity, incompatibility, and LIFO constraints. The objective function minimizes the sum of the molten iron waiting time of the loaded TPCs (associated with BA→RS requests) and the travel time of locomotives.

The contributions of this paper are as follows:

- We addressed a new problem, termed as the locomotive routing problem (LRP), in the steel industry. Two mathematical formulations are presented herein for this problem: a three-index formulation and a group-based formulation.
- To solve this practical problem, we designed an adaptive large neighborhood search (ALNS) algorithm. New removal operators (based on *block re-optimization*) and insertion operators were proposed, which can be beneficial for the diversification of neighborhood exploration and solution quality improvement. Moreover, a local search procedure, including adapted neighborhood structures and greedy acceptance criterion based on special characteristics of the LRP, was designed to further improve the solution quality.
- We also present extensive computational experiments based on actual data. The rule-based manual scheduling method, currently used by the steel company considered in this study, is compared with the ALNS algorithm. We also investigate the effectiveness of different ALNS components.

The remainder of this paper is organized as follows. The next section reviews the literature on related problems. In Section 3, the problem addressed in this paper is formally described. Also in this section, two mathematical formulations are introduced for the problem. Section 4 presents the ALNS algorithm. Section 5 reports the results of the computational study. Finally, Section 6 concludes the paper.

2. Literature Review

Despite its significance in practice, literature pertaining to molten iron transportation problems remains scarce. Tang et al. (2007) proposed a branch-and-price algorithm for the molten iron allocation problem,

which focused on allocating virtual molten iron batches (evaluated as TPCs) to the production plans of steelmaking. Huang et al. (2011) studied a similar problem with several production processes and designed a two-stage heuristic algorithm to solve it. Li et al. (2016a) proposed an improved discrete artificial bee colony algorithm to solve this scheduling problem with dynamic operation skipping features.

For the TPC scheduling problem, which determines the cyclic usage of TPCs, Liu & Wang (2015) studied a problem for designing optimal delivery routes to transport molten iron directly from the BFs to RSs via a fleet of TPCs. Both heuristic and exact algorithms were designed for a more complex TPC scheduling problem, where the pretreatment (desulfurization) before the unloading of the molten iron as well as the buffer zone for parking the loaded TPCs (before the pretreatment) were considered (Kletzander & Musliu, 2017; Geiger et al., 2019; Goldwaser & Schutt, 2018).

These previous studies assumed a sufficient number of locomotives to transport the TPCs. However, in most steel companies, the number of locomotives is limited. Kikuchi et al. (2008) proposed a heuristic algorithm for an integrated scheduling problem, including TPC scheduling, transfer request assignment, and locomotive route planning. Deng et al. (2011) designed a modified ant colony optimization algorithm for determining the optimal path for a locomotive in a steel company.

The transportation devices in the locomotive assignment problem (Piu et al., 2015; Xu et al., 2018) and the LRP are identical, and both these problems utilize railroad objects; however, these two problems themselves are considerably different. This is attributed to the characteristics of the locomotives and transported objects. Despite of the different origin and destination of locomotives, the incompatibility of requests, and the different capacity of locomotive for transporting empty and loaded requests, the LRP is a variant of the vehicle routing problem (VRP). More precisely, it is a variant of the pickup and delivery problem with time windows and LIFO loading constraints (PDPTWL). Hence, we focus on the review of transportation problems with loading constraints. Models and algorithms for the VRP and its variants, including exact and heuristic algorithms, can be found in Berbeglia et al. (2007); Parragh et al. (2008); Baldacci et al. (2008); Baldacci & Mingozzi (2009); Baldacci et al. (2012); Costa et al. (2019); Harbaoui Dridi et al. (2020).

Several works (Cordeau et al., 2010; Li et al., 2011; Pollaris et al., 2015; Wei et al., 2015; Veenstra et al., 2017b) have investigated the pickup and delivery traveling salesman problems with loading sequence constraints (TSPPDL). Cheang et al. (2012) studied the TSPPDL by considering multiple vehicles with unlimited capacity but limited total travel distance. A two-stage approach was designed to minimize the total distance and number of vehicles. Instead of the route length, Benavent et al. (2015) addressed the TSPPDL for multiple capacitated vehicles and limited route duration (PDPLT). Both exact and heuristic algorithms were proposed for solving this problem. The current state-of-art algorithm for solving the PDPLT is a learning-based metric algorithm designed by Peng et al. (2020).

Cherkesly et al. (2015a) addressed a PDPTWL that ignored the limitation on route duration of PDPLT but included time window constraints of the pickup and delivery operations. They proposed three branch-and-cut-and-price algorithms for the PDPTWL to optimally solve instances containing 75 requests, within one hour. To the best of our knowledge, the branch-and-cut algorithm designed by Alyasiry et al. (2019) for the PDPTWL is the current state-of-the-art exact method, capable of solving up to 125 requests. Considering the limitations of exact algorithms, heuristics were also designed to handle large-scale instances (Cherkesly et al., 2015b; Liu et al., 2019). Cherkesly et al. (2016) addressed the need for an extension of the PDPTWL, in which the vehicles have multiple stacks. They proposed two exact methods to solve instances with up to 75 requests and with 1-3 stacks to the optimum, within two hours. Additionally, by including the rehandling operation of requests that are onboard, the complexity of routing problems with LIFO constraints will increase, but a more flexible mode of transportation can be implemented (Battarra et al., 2010; Veenstra et al., 2017a,b; Hornstra et al., 2020; Cherkesly & Gschwind, 2022).

The routing of locomotives at industrial in-plant railroads is closely related to our study. Lübbecke & Zimmermann (2003) proposed a mixed integer formulation based on pre-generated patterns comprising requests served (picked) together by locomotives. They also introduced a set partitioning model and designed a branch-and-price algorithm to minimize the total working horizon of locomotives. Wang & Tang (2007) studied a locomotive scheduling problem with a nonlinear objective function. A locomotive can, at most, serve two TPCs simultaneously, and different request types cannot be transported together.

Here, we build on the research of Lübbecke & Zimmermann (2003) and Wang & Tang (2007). The LRP studied is an extension of the problems addressed in these two studies, but with more realistic characteristics. Specifically, we consider the following two new characteristics. First, the capacity of a locomotive for transporting the empty TPCs varies from two to four. Second, we consider the operations in the buffer area; this leads to additional types of pickup and delivery operations and more complicated hard time windows.

As shown by our computational experiments, the direct solution of the two IPs using a general MIP solver is only effective for small-scale LRP instances. In practice, an algorithm that is capable of computing high quality solutions in a short amount of time is needed for the LRP. The ALNS algorithm was proposed by Ropke & Pisinger (2006a) for solving PDPTW. The ALNS algorithm is based on destruction and reconstruction, performed using dynamically selected removal and insertion operators, respectively. This metaheuristic is effective in solving different kinds of routing problems such as, the pollution-routing problems (Demir et al., 2012; Franceschetti et al., 2016), the PDPTW with profits and reserved requests (Li et al., 2016b), the VRP with simultaneous pickup and delivery and handling costs (Hornstra et al., 2020), the VRPTW and delivery robots (Chen et al., 2021), and other variants of VRP (Pisinger & Ropke, 2007; Gendreau & Potvin, 2019). Motivated by the success of this algorithm in solving various routing problems, we adopt the ALNS algorithm to solve the LRP. According to the new characteristics of LRP, both adapted and new operators were proposed, as well as a local search procedure with adapted neighborhood structures and greedy acceptance criterion.

3. Problem description and mathematical formulations

The LRP can be formally described as follows. Let $\mathcal{R} = \mathcal{R}_E \cup \mathcal{R}_L$ be a set of transportation requests, where set $\mathcal{R}_E$ is the transfer of empty TPCs (i.e., BA→BF or RS→BA) and set $\mathcal{R}_L$ is the transfer of fully loaded TPCs (i.e., BF→BA or BA→RS). $\mathcal{R}_{L_1} \subset \mathcal{R}_L$ is the subset of transportation requests corresponding to the loaded TPCs from BA to RS.

For each request $r \in \mathcal{R}$, the pickup and delivery locations are given by the pair (r^+, r^-), respectively, where r^+ and r^- represent the possible locations associated with the BA, BFs, and RSs. Let $\mathcal{R}^+(\mathcal{R}^-)$ denote the corresponding pickup (delivery) location set of $\mathcal{R}$, where $\mathcal{R}^+ = \{r^+ | r \in \mathcal{R}\}(\mathcal{R}^- = \{r^- | r \in \mathcal{R}\})$. Let $\mathcal{R}_L^+$ $(\mathcal{R}_E^+)$ and $\mathcal{R}_L^-$ $(\mathcal{R}_E^-)$ be the sets of the pickup and delivery locations of $\mathcal{R}_L$ $(\mathcal{R}_E)$, respectively. Additionally, $\mathcal{R}_{L_1}^-$, a subset of $\mathcal{R}_L^-$, represents the set of loaded TPC (BA→RS) delivery locations. Each request BF→BA has an associated request, BA→RS, representing the combined transportation of a loaded TPC from the BF to the final RS. Additionally, each transportation request $r \in \mathcal{R}_{L_1}$ is associated with a parameter, tf_r, representing the expected completion time of the TPC loading operation at BF. The temperature drop of the molten iron is proportional to the difference between the arrival time at the RS (a decision variable) and the completion time tf_r. Each transportation request $r \in \mathcal{R}$ is also associated with the constant service times s_+^r and s_-^r and time windows $[e_+^r, l_+^r]$ and $[e_-^r, l_-^r]$ at the pickup and delivery locations, respectively. If a locomotive arrives at the pickup (delivery) location of request r before time e_+^r (e_-^r), it waits until time e_+^r (e_-^r).

To service the transportation requests, a set of locomotives, $\mathcal{L}$, are used. For each locomotive, $\ell \in \mathcal{L}$, a start location ℓ^+ (i.e., where it commences its service) and an end location ℓ^- (i.e., where it terminates its working day) is assigned. Let $\mathcal{L}^+$ $(\mathcal{L}^-)$ denote the set of locomotive origin (destination), where $\mathcal{L}^+ =$

$\{\ell^+|\ell \in \mathcal{L}\}$ ($\mathcal{L}^- = \{\ell^-|\ell \in \mathcal{L}\}$). For each locomotive $\ell \in \mathcal{L}$, a time window $[et_\ell, lt_\ell]$ is defined, representing the working period within the planning horizon $[0, T]$. The capacity of locomotives for transporting empty and loaded requests c_e and c_l equals to four and two, respectively.

The underlying transportation network is represented by a graph $G = (V, A)$ where the set of vertices $V = \{r^+, r^-\}_{r\in\mathcal{R}} \cup \{\ell^+, \ell^-\}_{\ell\in\mathcal{L}}$ represents the different locations and the set of arcs A represents the possible links among the different vertices in V. Each arc, $(i, j) \in A$, is associated with a travel time t_{ij}. Let T_j be the visiting time at the destination of an arc (i, j), then defining a function $f_{ij}(T_j)$ to compute the arc cost (i, j) is as follows.

$$f_{ij}(T_j) = \begin{cases} t_{ij} + (T_j - tf_r), & if\ j = r^- \in \mathcal{R}^-_{L_1}, \\ t_{ij}, & otherwise. \end{cases} \tag{1}$$

Equation (1) express that, if the destination of an arc corresponds to the delivery location of transportation request from BA to RS, the cost of an arc equals the sum of the locomotive travel time and the molten iron waiting time. Otherwise, the cost of an arc equals the locomotive travel time.

We define P, a *locomotive route* or simply a route for a locomotive $\ell \in \mathcal{L}$, as a path in G starting from vertex ℓ^+ at et_ℓ and ending at vertex ℓ^- no later than time lt_ℓ and servicing a subset $P(\mathcal{R}) \subseteq \mathcal{R}$ of requests within its pickup and delivery time windows. A locomotive route is divided into *empty* locomotive transfers, where the locomotive moves between two locations without TPCs attached to it, and *loaded* transfers, where the locomotive transports TPCs. The following constraints must be respected by a route P:

- **Capacity constraints**: A locomotive can simultaneously service two and four requests at the most from set $\mathcal{R}_L$ and set $\mathcal{R}_E$, respectively.
- **Request incompatibility constraints**: Due to stability constraints, the empty and loaded TPCs or requests cannot be serviced simultaneously.
- **LIFO loading constraints**: To avoid unnecessary time or risk related to rearranging the TPCs at the delivery locations, the last TPC picked up by a locomotive need not be moved until it reaches its destination. Therefore, it will be the first TPC to be delivered along the path.

The cost of a route is defined as the sum of the arc costs traversed by the route. The LRP calls for the design of $|\mathcal{L}|$ routes with the minimum total cost, including one route for each locomotive, such that each transportation request in $\mathcal{R}$ is served by exactly one route.

Fig.3 illustrates an example of a LRP solution with three locomotives and six requests. The number associated with each arc is the traveling time between two nodes. $([e, l], tf_r, pos, typ, obj)$, a five-dimensional vector, is used to illustrate the node information of a request (pickup and delivery nodes) or a locomotive (origin and destination nodes). $[e, l]$ represents the time window of a node, and a symbol "*" denotes that the node is not a request from BA to RS. Term pos denotes its location, term typ indicates the node type, where "+" and "-" are the origin and destination of a locomotive or request, respectively. Term obj is the relative objective value after the visit of a node.

The routes of ℓ_1 and ℓ_3 are loaded locomotive transfers while ℓ_2 is an empty locomotive transfer. The LIFO rule is used for requests r_3 and r_4.The transportation of loaded request r_1 from BF to BA will only increase the locomotive travel time. Nevertheless, besides the locomotive travel time (10 minutes) of loaded request r_6 delivered from BA to RS, an additional contribution to the obj w.r.t. the molten iron waiting time (58 minutes) is also included.

We introduce two mathematical formulations for the LRP: a three-index formulation and a group-based formulation. In the group-based formulation, the LIFO rule is implicitly considered in the definition of the decision variables and infeasible request combinations are also implicitly eliminated. However, the number of decision variables increases dramatically as the number of requests increases. Therefore, we also propose

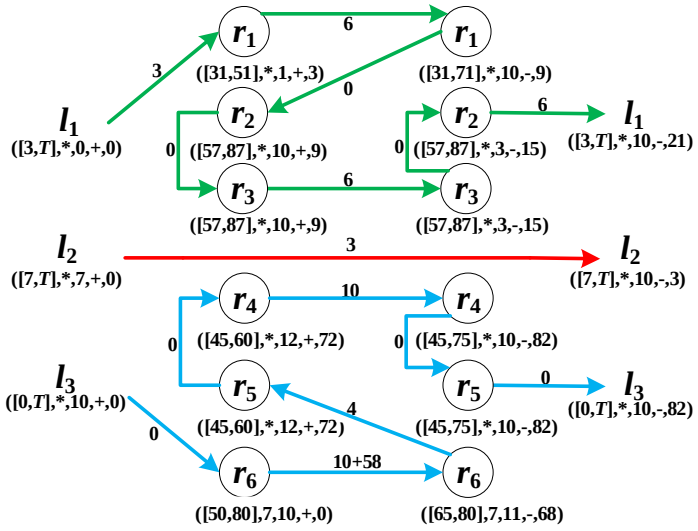

Fig. 3. Routes of a feasible solution.

an alternative arc-based three-index formulation. For the sake of the space, below we report the three-index formulation whereas details of the group-based formulation are given in the Online Supplement.

Let n be the number of pickup locations, where $n = |\mathcal{R}^+|$. In the formulation, coefficient η is a large constant. The service time of a node $i \in V$ is denoted by s_i, which equals 0 if $i \in \mathcal{L}^+ \cup \mathcal{L}^-$; otherwise, it equals 2. Coefficient q_i represents the quantity of node $i \in V$, which equals 2, if $i \in \mathcal{R}_L^+$; -2, if $i \in \mathcal{R}_L^-$; 1, if $i \in \mathcal{R}_E^+$; -1, if $i \in \mathcal{R}_E^-$; and 0, if $i \in \mathcal{L}^+ \cup \mathcal{L}^-$. The decision variables used in the three-index formulation are as follows:

T_i: the service start time of the locomotive at node $i \in V$;
$x_{ij}^{\ell} \in \{0, 1\}$: equals 1 if locomotive ℓ travels along arc (i, j) and 0, otherwise;
y_i^{ℓ} : the load of locomotive ℓ after visiting node $i \in V$.

The three-index formulation is as follows:

$$min \sum_{\ell \in \mathcal{L}} \sum_{(i,j) \in A} x_{ij}^{\ell} f_{ij}(T_j) \tag{2}$$

$$s.t. \sum_{i \in \mathcal{R}^+ \cup \ell^-} x_{\ell^+ i}^{\ell} = 1, \forall \ell \in \mathcal{L} \tag{3}$$

$$\sum_{i \in \mathcal{R}^- \cup \ell^+} x_{i\ell^-}^{\ell} = 1, \forall \ell \in \mathcal{L} \tag{4}$$

$$\sum_{\ell \in \mathcal{L}} \sum_{(i,j) \in A} x_{ij}^{\ell} = 1, \forall i \in \mathcal{R}^+ \tag{5}$$

$$\sum_{(i,j) \in A} x_{ij}^{\ell} - \sum_{(i+n,j) \in A} x_{i+n,j}^{\ell} = 0, \forall i \in \mathcal{R}^+, \ell \in \mathcal{L} \tag{6}$$

$$\sum_{(i,j) \in A} x_{ij}^{\ell} - \sum_{(j,i) \in A} x_{ji}^{\ell} = 0, \forall i \in \mathcal{R}^+ \cup \mathcal{R}^-, \ell \in \mathcal{L} \tag{7}$$

$$\sum_{j \in \mathcal{R}^-} x_{ij}^{\ell} - x_{i,i+n}^{\ell} = 0, \forall i \in \mathcal{R}^+, \ell \in \mathcal{L} \tag{8}$$

$$x_{ij}^{\ell} - x_{j+n,i+n}^{\ell} = 0, \forall i, j \in \mathcal{R}^{+}, \ell \in \mathcal{L} \tag{9}$$

$$\sum_{j \in \mathcal{R}_L^{+}} x_{ij}^{\ell} + \sum_{j \in \mathcal{R}_L^{+}} x_{ji}^{\ell} = 0, \forall i \in \mathcal{R}_E^{+}, \ell \in \mathcal{L} \tag{10}$$

$$T_i + s_i + t_{ij} + (x_{ij}^{l} - 1)\eta \leq T_j, \forall (i, j) \in A, \ell \in \mathcal{L} \tag{11}$$

$$y_i^{\ell} - q_i = y_{i+n}^{\ell}, \forall i \in \mathcal{R}^{+}, \ell \in \mathcal{L} \tag{12}$$

$$y_i^{\ell} + q_j + (x_{ij}^{\ell} - 1)\eta \leq y_j^{\ell}, \forall (i, j) \in A, \ell \in \mathcal{L} \tag{13}$$

$$y_{\ell^{+}}^{\ell} + y_{\ell^{-}}^{\ell} = 0, \forall \ell \in \mathcal{L} \tag{14}$$

$$q_i \leq y_i^{\ell} \leq 4, \forall i \in \mathcal{R}^{+}, \ell \in \mathcal{L} \tag{15}$$

$$0 \leq y_i^{\ell} \leq 4 + q_i, \forall i \in \mathcal{R}^{-}, \ell \in \mathcal{L} \tag{16}$$

$$e_i \leq T_i \leq l_i, \forall i \in V. \tag{17}$$

The objective function (2) minimizes the sum of the molten iron waiting time and the locomotive travel time, whereby the calculation of $f_{ij}(T_j)$ follows equation (1). Constraints (3) and (4) restrict the number and type of arcs starting from the origin and reaching the destination of every locomotive. Constraints (5) ensure that each pickup location of a request is visited by a locomotive exactly once. Additionally, constraints (6) guarantee that the corresponding delivery location is visited by the same locomotive. Constraints (7) ensure the flow conservation of the nodes in $\mathcal{R}^{+} \cup \mathcal{R}^{-}$. Constraints (8) and (9) define the LIFO rule: (8) state that only the delivery location of a request can be visited directly after its pickup location, and (9) impose that the visiting order of the two delivery locations is the reverse of the pickup locations. Constraints (10) guarantee that requests of different type cannot be transported together. Constraints (11) impose the service start times consistency for two successive nodes in a route. Equations (12) link the load of the locomotive at the pickup and delivery nodes of a request. Constraints (13) impose the load consistency between consecutive nodes visited by the locomotives. Constraints (14) impose that the load quantity at the origin and destination of every locomotive is zero. The locomotive capacity constraint are then imposed by constraints (15)-(16). Constraints (17) are the time window constraints of each node. Note that constraints (10) can be ignored by eliminating two types of arcs $(i, j) \in A$. The first type of arc satisfies that $i \in \mathcal{R}_E^{+}$ and $j \in \mathcal{R}_L^{+}$ or $i \in \mathcal{R}_L^{+}$ and $j \in \mathcal{R}_E^{+}$. The second type of arc satisfies $i \in \mathcal{R}_E^{-}$ and $j \in \mathcal{R}_L^{-}$ or $i \in \mathcal{R}_L^{-}$ and $j \in \mathcal{R}_E^{-}$.

4. An ALNS algorithm for the LRP

The framework of the proposed ALNS algorithm is presented in Algorithm 1. The ALNS algorithm starts with an initial solution that is generated through the method introduced in Section 4.1. Then a new solution is generated by using selected removal and insertion operators. Those operators are selected based on its weights, which are adjusted based on its past performance as shown in Section 4.2. Considering the LIFO rule, incompatibility of requests, and the variable locomotive capacity for loaded and empty requests, a new method for insertion feasibility testing is introduced in Section 4.3. The details of removal and insertion operators are presented in Section 4.4. When a new solution is generated, whether to accept it or not should follow the criterion described in Section 4.5. For further improvement, a local search procedure (see Section 4.6) is executed at the end of each iteration.

4.1. Initialization

The method used to create the initial solution is a combination of two heuristics. All requests are unserved at the beginning of the proposed algorithm, and regret insertion operators (see Section 4.4.1 and

Algorithm 1: Framework of the proposed ALNS algorithm

Input: removal operators $\mathcal{D}$, insertion operators $\mathcal{I}$, initial temperature T^{org}, cooling rate cr
Output: A solution X_{best}

Generate an initial solution X_{cur} by using the method introduced in Section 4.1
Initialize probability for each $d \in \mathcal{D}$ and $\iota \in \mathcal{I}$, $X_{new} \leftarrow X_{best} \leftarrow X_{cur}$, $T^{cur} \leftarrow T^{org}$
Let θ be the iteration number counter initialized as $\theta \leftarrow 1$
Let α be the segment iteration number counter initialized as $\alpha \leftarrow 1$
while $\theta \leq NI$ **do**
    Select a removal operator $d \in \mathcal{D}$ to destroy X_{cur}(Section 4.4.1)
    Select an insertion operator $\iota \in \mathcal{I}$ to reinsert requests in L to get X_{new}(Section 4.4.2)
    if $obj(X_{new}) < obj(X_{cur})$ *(Section 4.5)* **then**
        $X_{cur} \leftarrow X_{new}$
        if $obj(X_{new}) < obj(X_{best})$ **then** $X_{best} \leftarrow X_{new}$;
    else
        Let $\zeta \leftarrow e^{-(obj(X_{new})-obj(X_{cur}))/T^{cur}}$
        Generate a random number $\xi \in [0, 1]$
        if $\xi < \zeta$ **then** $X_{cur} \leftarrow X_{new}$;
    end
    Update the score and usage times of operators selected
    if $\alpha = NIS$ **then**
        Update the probabilities using adaptive weight adjustment procedure(Section 4.2)
        Execute the local search procedure(Section 4.6)
        Reinitialize α as $\alpha \leftarrow 1$
    end
    $T^{cur} \leftarrow T^{cur} cr$
    $\theta \leftarrow \theta + 1, \alpha \leftarrow \alpha + 1$
end

Section 4.4.2) are used to construct the candidate solutions. Subsequently, the local search procedure presented in Section 4.6 is applied to further improve the quality of selected solution, which is the best one among the candidate solutions.

4.2. Adaptive score and weight adjustment

Even though the removal and insertion operators are selected according to the roulette wheel selection principle, an operator with a large weight is more likely to be chosen. If the operator weights remain fixed until the end of the iterative process, certain operators may barely be used. Therefore, the weight adjustment method is employed as follows. Let $d \in \mathcal{D}$ be a removal operator and $\iota \in \mathcal{I}$ be an insertion operator, where $\mathcal{D}$ and $\mathcal{I}$ are the sets of the removal and insertion operators, respectively. Let ω_i^t be the weight value that is reset after a fixed number of iterations (NIS), i.e., a segment of the algorithm.

Three parameters, π_1, π_2, and π_3, denote the different score increments of the removal and insertion operators in accordance with the new solution quality. If the new solution is better than the global best solution, π_1 is added to the score of the two operators used. If the new solution is better than the current solution, the score increment is π_2. If the new solution is worse than the current one but accepted, the score increment is π_3. Parameter ω_i^t denotes the weight of operator i used during segment t and initialized as 1; ω_i^t is set according to $\omega_i^{t+1} = \omega_i^t(1-\rho) + \rho\kappa_i/\gamma_i$ at the beginning of segment $t+1$, where $\rho = 0.1$ is the reaction factor of the roulette wheel selection algorithm. Parameters κ_i and γ_i are the score and usage times, respectively, of operator i in the current finished segment. The probabilities $\mathcal{P}_i^t$ of a removal or insertion operator being used in the operator selection in the t-th segment are calculated as follows:

$$\mathcal{P}_i^t = \omega_i^t / \sum_{d=1}^{|\mathcal{D}|} \omega_d^t \quad for \;\; i \in \mathcal{D}, \quad \mathcal{P}_i^t = \omega_i^t / \sum_{\iota=1}^{|\mathcal{I}|} \omega_\iota^t \quad for \;\; i \in \mathcal{I}.$$

4.3. Feasibility of request insertion

Inserting a request r into a route includes the insertion of r^+ and r^-. According to the LIFO rule, the candidate locations (*Case* 1-5) of inserting a request r in a route are shown in Fig.4. Request r^- cannot be inserted directly after r^+, when r^+ is between two pickup nodes (*Case* 3). Otherwise, the location after r^+ will be taken into consideration. Three more aspects must be checked when inserting r : the violation of time windows, locomotive capacity, and incompatibility of requests.

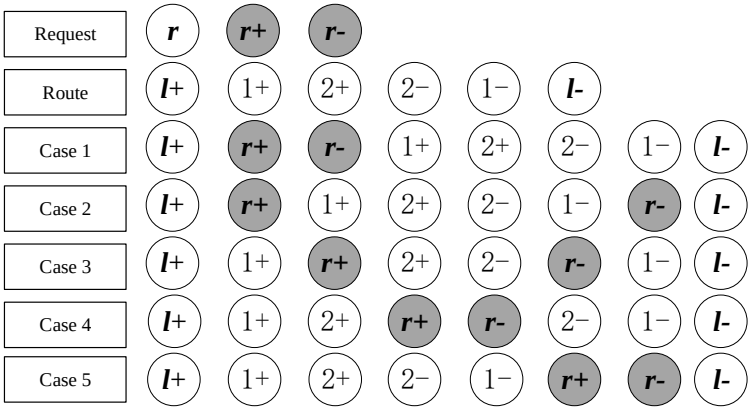

Fig. 4. Candidate locations for request insertion.

Algorithm 2: Feasibility check of request insertion

Input: Request $r \in \mathcal{R}_L(\mathcal{R}_E)$, New route of a locomotive ℓ
Output: Insertion Feasibility

I *Calculate m_1, m_2, and m_3*
if $m_1 = 0$ **then**
  if r_p^+ *is* ℓ^+ **then**
    if $m_3 = 0$ **then** Goto II;
    else if $m_2 < c_l(c_e)$ & $m_3 = 2m_2$ & $r_s^+ \in \mathcal{R}_L^+(\mathcal{R}_E^+)$ **then** Goto II;
    else *Return false*;
  else
    if $m_3 = 0$ & r_s^- *is* ℓ^- **then** Goto II;
    else if $m_3 = 0$ & $r_s^- \in \mathcal{R}^+$ **then** Goto II;
    else if $m_2 < c_l(c_e)$ & $m_3 = 2m_2$ & $r_s^+ \in \mathcal{R}_L^+(\mathcal{R}_E^+)$ **then** Goto II;
    else *Return false*;
  end
else
  if $r_p^+ \in \mathcal{R}_L^+(\mathcal{R}_E^+)$ & $m_1 + m_2 < c_l(c_e)$ **then**
    if $m_2 + m_3 = 0$ **then** Goto II;
    else if $m_2 > 0$ & $m_3 = 2m_2$ **then** Goto II;
    else *Return false*;
  else *Return false*;
end
II *Check the time window feasibility as follows*
if *No violation of time windows* **then** *Return true* ;
else *Return false*;

In the new route, let m_1 and m_2 be the number of pickup requests directly before and after r^+, respectively. Let m_3 denote the total number of pickup and delivery operations between r^+ and r^-. Let r_p^+ (r_p^-) and r_s^+ (r_s^-) denote the predecessor and successor of r^+ (r^-), respectively. Clearly, the location of r^- in the new route must be after r^+ and before ℓ^-. Details pertaining to the insertion feasibility check are shown in Algorithm 2.

4.4. Removal and insertion operators

There are nine removal operators and seven insertion operators can be used for the destruction and reconstruction of the ALNS algorithm. In each iteration, a removal operator is selected to destroy the current solution by removing q requests. Those removed requests are added into the removal list L. After the destruction, an insertion operators is used to repair the destroyed solution. Each time, a request in L will be chosen and reinserted based on the selected insertion operator. This process will continue until L is empty.

4.4.1. Removal operators

A total of nine request removal operators are explained in this section. The first five of removal operators are commonly used in the literature (Ropke & Pisinger, 2006a,b; Pisinger & Ropke, 2007; Demir et al., 2012; Sun et al., 2020b). A new component is added in *shaw removal* operator, considering the incompatibility of empty and loaded requests. We define a new function to calculate the adjacent cost of a request using the *historical knowledge removal* operator. In the LRP, sometimes transporting requests together may produce a better solution (see the example in Section 4.6.2). Notably, only requests of the same type and with small time window differences that can be transported together. Hence, we introduce

four new operators to perform *block re-optimization* by removing requests from the same time block. The specific descriptions of operators are as follows.

Random Removal (RR) All the requests are sorted randomly, and the first q requests are selected. Those selected requests will be removed from the current solution and added to L. This operator is proposed to diversify the search.

Worst Removal (WR) This operator relocates the request that causes the largest objective value decrement of the current solution, which may result in a potentially superior new solution. Let f_r and f_{-r} be the objective values of the current solution and its corresponding solution after the removal of request r, $r \in \mathcal{R}$, respectively. Let $f^* = max\{f_r - f_{-r} | r \in \mathcal{R}\}$ denote the largest objective value reduction. This operator calculates f^* and adds the corresponding request to the removal list until $|L| = q$. Note that f^* must be recomputed before a new round of removal request selection.

Single Route Removal (SRR) In order to obtain a significant variant of the current solution, this operator removes all requests belonging to a randomly chosen route.

Shaw Removal (SR) The purpose of this operator is to remove requests with similarities. The relatedness $R(r, r')$, used for measuring the similarity between two requests, i.e., r and r', is calculated as follows: $R(r, r') = \phi_1(|e^r_+ - e^{r'}_+| + |e^r_- - e^{r'}_-|) + \phi_2(|l^r_+ - l^{r'}_+| + |l^r_- - l^{r'}_-|) + \phi_3((|T_{r^+} - T_{r'^+}| + |T_{r^-} - T_{r'^-}|)) + \phi_4(t_{r^+r'^+} + t_{r^-r'^-}) + \phi_5\mathcal{B}$, where ϕ_1, ϕ_2, ϕ_3, ϕ_4, and ϕ_5 are the weights of different components of relatedness. Coefficient $\mathcal{B}$ equals one, if r and r' are of different types; otherwise, it equals zero. The remaining components of relatedness are normalized such that $R(r, r') \in [0, \sum_{i=1}^{5} \phi_i]$. If this operator is chosen, a request r is first selected randomly and inserted into L. Thereafter, the last added request $r' \in L$ is selected; the request in the current solution has minimum relatedness to r' is removed and added to L, until $|L| = q$.

Historical Knowledge Removal (HKR) This operator intends to move requests according to the cumulative historical information based on the previous iterations. Let $C_r = t_{r^+_p r^+} + t_{r^+ r^+_s} + t_{r^-_p r^-} + t_{r^- r^-_s}$ be the adjacent cost of request r (the sum of the adjacent costs of r^+ and r^-), where subscripts p and s indicate the predecessor and successor of r^+ or r^-, respectively. Let C^{opt}_r be the smallest recorded adjacent cost of r, which is updated when any change in its linked edges leads to a reduced adjacent cost. A request r with maximum adjacent cost deviation from its best record, i.e., $r = argmax_{r \in \mathcal{R}}\{C_r - C^{opt}_r\}$, is removed and added to L.

Time Oriented Removal (TOR) Let *length* denote a time length used for the request. A request r is randomly selected in this operator. If request r' satisfies $|e^r_+ - e^{r'}_+| \leq length$ or $|l^r_+ - l^{r'}_+| \leq length$, remove r' and add it to L. Continue this process until $|L| = q$. If $|L| < q$ and no candidate request exists, enlarge *length* as $length = length + 10$ and perform the selection again.

Cost Oriented Removal (COR) This operator combines the *WR* and *TOR* operators. The request having the largest objective value decrement of the current solution, is first added to L. The remaining $q-1$ requests are selected by the method used in *TOR*.

Delay Oriented Removal (DOR) This operator stipulates the removal of a delayed request and other requests with similar service start time. A route is randomly selected and then the first request r that does not commence its service at the earliest time is removed and added to L. If L remains empty after the searching of all routes, a request will be randomly removed and added to L. The time interval $[\mathcal{T}_{lb}, \mathcal{T}_{ub}]$ is calculated according to request r, where $\mathcal{T}_{lb} = max\{T_{r^+} - length, 0\}$ and $\mathcal{T}_{ub} = min\{T_{r^+} + length, T\}$. Requests that are picked within the time interval $[\mathcal{T}_{lb}, \mathcal{T}_{ub}]$ are added to L. If $|L| \geq q$, stop and choose the first q requests in L for removal. Otherwise, enlarge *length* as $length = length + 10$ and perform the selection again.

Extend Delay Oriented Removal (EDOR) This operator is similar to the previous one. However, the content first added to L does not only include the delayed request but also the preceding group of requests

served, as shown in Fig.5. Suppose r is the delayed request that has been selected, its preceding group $(1+, 2+, 2-, 1-)$ will be removed and all requests in this group are added in L at the same time. This operator can be treated as a supplement to *DOR* that emphasizes the rescheduling of the delayed requests and their predecessors.

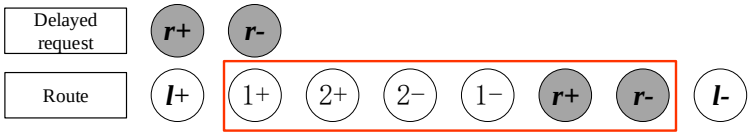

Fig. 5. Request selection example of extend delay oriented removal operator.

4.4.2. Insertion operators

After the execution of selected removal operator, an insertion operator is used to repair the destroyed solution. Here, we present seven operators based on previous studies. The first six operators are commonly used in the literature (Ropke & Pisinger, 2006a; Franceschetti et al., 2016; Hornstra et al., 2020). The last operator is inspired by the *greedy insertion* operator.

Greedy Insertion (GI) This operator entails a simple iterative process. In each iteration, the objective value increment of inserting a request into every feasible position is calculated for requests in L. Then the insertion of a request corresponding to the smallest objective value increment will be executed.

Regret Insertion (RI) This is a class of k-regret insertion operators, where $k \in \{2, 3, 4, 7\}$ in this study. The different k values represent the number of routes with the first k minimum insertion cost. The *RI* takes the corresponding objective value increments of the different insertion positions into account to avoid myopic insertion.

Random Sequence Insertion (RSI) This operator randomly selects a request from L which is then inserted into the best position. This operator is quite faster than the *GI* operator because it only needs to compute the best insertion position for the selected request.

First Fit Insertion (FFI) This operator aims to expand the neighborhood search and it is beneficial to diversify the search over the search space. The difference between this operator and operator *GI* is that, for each request in L, the exploration of insertion locations stops whenever the insertion is feasible. Then, the request in L having the smallest objective value increment is inserted into the corresponding location.

Insertion operators are randomized by applying noise to their objective functions. This is realized by adding a term $t_{ij}^{max} w\epsilon$ to the insertion objective value, where t_{ij}^{max} is the largest travel time of all the arcs, w is a randomly generated number within $[-1, 1]$, and ϵ is a noise-controlling parameter. Since the selected request is only inserted into a feasible position, we will not apply noise to FFI.

4.5. Acceptance and stopping criteria

Similarly to the corresponding component in standard ALNS algorithm, the acceptance criterion used in our algorithm employs features of Simulated Annealing (SA) algorithms to avoid the trap of local optimum. Let T^{cur} denote the current temperature, which is initialized by the original temperature T^{org}. Let cr denote the cooling rate, where $cr \in (0, 1)$. T^{cur} and cr are settings that make it possible to accept a solution that is worse than the current one. As T^{cur} decreases according to cr, $T^{cur} = T^{cur} * cr$, the probability of accepting a worse solution also decreases. Finally, when $T^{cur} = 0$, only the solution without a deterioration in the objective value can be accepted. Let X_{cur} and X_{new} denote the current and new (not generated previously) solutions, respectively. Let $obj(X)$ denote the objective value of solution X. If $obj(X_{new}) < obj(X_{cur})$, X_{new}

is accepted; otherwise, X_{new} is accepted with the probability $e^{-(obj(X_{new})-obj(X_{cur}))/T^{cur}}$. This algorithm stops after a fixed number of iterations (NI).

4.6. Local Search

The proposed local search procedure consists of a greedy acceptance criterion and six neighborhood structures $\mathcal{N}_i$, where $i \in \{1, \ldots, 6\}$. $\mathcal{N}_1 - \mathcal{N}_4$ are adapted from the neighborhood structures introduced by Cherkesly et al. (2015b). Those neighborhood structures have been redesigned according to the special LIFO constraints of the LRP. Neighborhood structures $\mathcal{N}_5$ and $\mathcal{N}_6$ are adapted from the $2 - opt^*$ exchange heuristic proposed by Potvin & Rousseau (1995). To speed up the computation, the local search procedure is used only at the end of each segment of iterations of the ALNS algorithm.

Souza et al. (2010) determined that a fixed exploration order of neighborhood structures cannot always be effective for instances of different types. Enhanced results can be produced by a random neighborhood structure exploration sequence (Subramanian et al., 2010), i.e., the randomized variable neighborhood descent (RVND). In the RVND, an index set $\mathcal{NL}$ is initialized as $\{2, \ldots, 6\}$. Then, a neighborhood $\mathcal{N}_i$ is randomly selected and explored, where $i \in \{2, \ldots, 6\}$. Following the searching for $\mathcal{N}_i$, if the resulting solution is better than the current one, then $\mathcal{N}_1$ of the new solution is searched and reset $\mathcal{NL} = \{2, \ldots, 6\}$. Otherwise, i is eliminated from $\mathcal{NL}$ and another element of $\mathcal{NL}$ is selected randomly. The RVND is terminated when $\mathcal{NL} = \emptyset$.

4.6.1. Intra-route neighborhood structure

This *intra-route* neighborhood exploits the relocation of a single request to reduce the objective value of every route. According to the special LIFO constraint, the relocation of a pickup node and the related delivery node cannot use the relocation moves introduced by Cherkesly et al. (2015b) directly.

Intra-route Single Request Relocation (SRR) Searching this neighborhood aims to improve the request scheduling for every route via request relocation. The relocation of a request r includes removing the request from current route and reinserting it into the best position. For each candidate location of $r+$, there may exist more than one candidate locations of $r-$ which should be checked, as shown in Fig.4. The best relocation of requests in a route are recorded and, among them, the relocation corresponding to the largest objective value decrement will be implemented. The search will continue until no objective value reduction can be found by relocating requests for each route.

4.6.2. Inter-route neighborhood structures

There are five *inter-route* neighborhood structures, the relocation of a single request or group, the relocation of multiple groups, the partial routes cross, and the route reassignment. The following two acceptance criteria are frequently used: the *first improvement* and *best improvement*. The first one stops the search as soon as a better solution is found, whereas the second explores the entire neighborhood and accepts the best neighbor. Let RVND-I and RVND-II denote the RVND algorithms associated with the two aforementioned criteria. If we choose *best improvement* as the acceptance criterion for an *inter-route* neighborhood search, useful information will be dropped when several feasible improvements exist after searching $\mathcal{N}_i$, $i \in \{2, \ldots, 5\}$. However, a better solution may not be generated by accepting all those moves w.r.t those neighbors.

Let $\mathcal{R} = \{r_1, r_2\}$ and $\mathcal{L} = \{\ell_1, \ell_2\}$ denote the request and locomotive sets, respectively. To simplify the description, we assume that r_1 and r_2 are identical empty requests and ℓ_1 and ℓ_2 are identical locomotives. The current solution consists of transporting r_1 by ℓ_1 and r_2 by ℓ_2, respectively. Six conditions are satisfied: (1) $et_\ell + t_{\ell^+ r^+} < e^r_+, r \in \mathcal{R}, \ell \in \mathcal{L}$; (2) $e^r_+ + 2s_{r^+} < l^r_+, r \in \mathcal{R}, \ell \in \mathcal{L}$; (3) $l^r_+ + t_{r^+ r^-} < e^r_-, r \in \mathcal{R}, \ell \in \mathcal{L}$; (4) $e^r_- + 2s_{r^-} < l^r_-, r \in \mathcal{R}, \ell \in \mathcal{L}$; (5) $l^r_- + 2s_{r^-} + t_{r^- \ell^-} < lt_\ell, r \in \mathcal{R}, \ell \in \mathcal{L}$; and (6) $t_{\ell^+_1 r^+_1} + t_{r^+_1 r^-_1} + t_{r^-_1 \ell^-_1} - t_{\ell^+_1 \ell^-_1} > 0$. (1)-(5) are

sufficient conditions that ensure the feasibility of transporting r_1 and r_2 together. Condition (6) ensures that transporting r_1 and r_2 together will result in a better solution than the current one. Clearly, if r_1 is removed from ℓ_1 and reinserted int ℓ_2, there are two best neighbors. The first one is $(\ell_1+, r_1+, r_2+, r_2-, r_1-, \ell_1-)$ and (ℓ_2+, ℓ_2-), while the other one is to transport r_1 and r_2 in reverse order. Similarly, another two best neighbors can be generated through the relocation of r_2. If we accept two moves corresponding to the best neighbor of r_1 and r_2, the new solution will transport r_1 by ℓ_2 and r_2 by ℓ_1. However, the new solution has an equal objective value when compared to the current solution.

To avoid this situation, a greedy acceptance criterion as shown in Algorithm 3 is used to generate a new solution for $\mathcal{N}_i, i \in \{2, \ldots, 5\}$. Let $m_{\ell\ell'}$ denote the objective value variation of a neighbor corresponding to the best move between ℓ and ℓ', where $m_{\ell\ell'} \in \mathcal{M}_{|\mathcal{L}|\times|\mathcal{L}|}$ and $\ell, \ell' \in \mathcal{L}$. $m_{\ell\ell'}$ is initialized as $+\infty$ if $\ell \neq \ell'$, and 0 if $\ell = \ell'$. A locomotive set $\mathcal{L}'$ is initialized as $\mathcal{L}$. Each time, the smallest $m_{\ell\ell'} \in \mathcal{M}_{|\mathcal{L}|\times|\mathcal{L}|}$ is selected, where $\ell, \ell' \in \mathcal{L}'$. Then, the move corresponding to the selected neighbor is executed and ℓ, ℓ' are removed from $\mathcal{L}'$. This process will continue until $\mathcal{L}'$ is empty.

Algorithm 3: Greedy acceptance criterion

Input: Solution X, one of $\mathcal{N}_i, i \in \{2, \ldots, 5\}$
Output: New solution X_{new}

Initialize $X_{new} \leftarrow X$, $\mathcal{L}' \leftarrow \mathcal{L}$, $m_{\ell\ell'} \leftarrow +\infty$, $m_{\ell\ell} \leftarrow 0, \ell, \ell' \in \mathcal{L}$
Explore $\mathcal{N}_i$ *of* X *and update* $\mathcal{M}_{|\mathcal{L}|\times|\mathcal{L}|}$
while $|\mathcal{L}'| > 0$ **do**
    Find the smallest $m_{\ell\ell'}$ *in* $\mathcal{M}_{|\mathcal{L}|\times|\mathcal{L}|}$, $\ell, \ell' \in \mathcal{L}'$
    Execute the corresponded operation of $m_{\ell\ell'}$ *on* X_{new}
    $m_{\ell\ell'} \leftarrow +\infty$
    if $\ell = \ell'$ **then** $\mathcal{L}' \leftarrow \mathcal{L}'/\{\ell\}$;
    else $\mathcal{L}' \leftarrow \mathcal{L}'/\{\ell, \ell'\}$;
end

Inter-route Single Request Relocation (ISRR) An exploration of this neighborhood aims to decrease the objective value of the current solution via the relocation of requests. Unlike the SRR, the removed request is reinserted into another route. For all requests served by locomotive ℓ, the relocation of it from ℓ to ℓ' are checked, where $\ell, \ell' \in \mathcal{L}$. Then, the best relocation is recorded as a candidate neighbor and the $\mathcal{M}_{|\mathcal{L}|\times|\mathcal{L}|}$ is updated accordingly. The new solution can be generated through Algorithm 3.

Inter-route Single Group Relocation (ISGR) This neighborhood is similar to the former one, except that it works on a group of requests that are transported together in the current solution. Fig.6(a) illustrates the relocation of a group $(2+, 3+, 2-, 3-)$. Besides, the insertion of selected group must not destroy any group in new route. Since every sub-path corresponding to a given group of requests has determined in current solution, the $ISGR$ neighborhood operation can save the time required to check the feasibility of the LIFO rule, locomotive capacity, and incompatibility of requests. This can greatly improve the search efficiency at the cost of losing partial search space.

Inter-route Multi-group Relocation (IMGR) The direct exchange of two groups in two routes will mostly lead to infeasible solution. Therefore, this neighborhood entails removing one group from two routes and relocating each group into the best position of another route. Fig.6(b) provides an example of the exchange of two groups $(2+, 5+, 5-, 2-)$ and $(3+, 3-)$ as well as the relocation of $(2+, 5+, 5-, 2-)$ in new routes. As mentioned above, when compared with using requests as operation objects, the search space is reduced to enable a reduction in computation time.

Inter-route Cross (IC) If we use the $2-opt^*$ exchange heuristic directly, the large number of candidate partial paths will result in a considerably long computation time. Moreover, most exchanges will be infea-

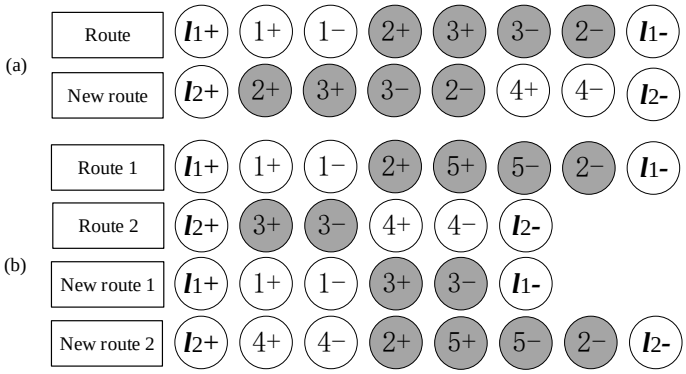

Fig. 6. Group relocation examples of *inter-route* neighborhood structures.

sible, owing to the violation of transportation constraints. Hence, we only check a restricted arc set of each route, i.e., the arcs from a delivery node to a pickup node. Two routes corresponding to ℓ and ℓ' are selected ($\ell, \ell' \in \mathcal{L}$), each of them is broken into two sub-paths by removing an arc. The broken four sub-paths are recombined as two new routes and then the destinations of two new routes are exchanged. As shown in Fig.7, two arcs $(1-, 2+)$ and $(4-, 5+)$ are first selected and removed. Then two sub-paths $(2+, \ldots, l_1-)$ and $(5+, \ldots, l_2-)$ are exchanged. Finally the end of the routes are exchanged to make sure that the origin and destination of each route belong to the same locomotive.

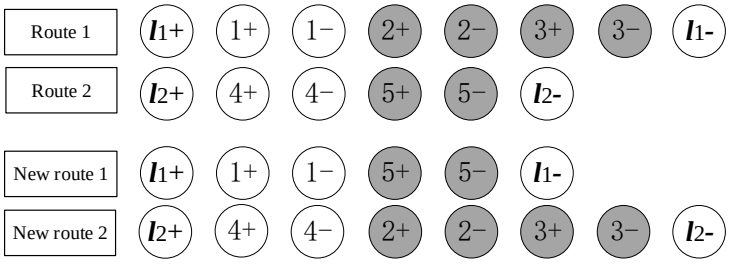

Fig. 7. Example of the *inter − route cross* neighborhood structure.

Inter-route Route Reassignment (IRR) The aim of this operator is to redefine the assignment of locomotives to the routes of a solution to reduce the total solution cost. Let p be the number of routes composing the solution and q be the number of locomotives. For each route i, $i = 1, \ldots, p$, and each potential locomotive j, $j = 1, \ldots, q$, we compute the route cost, say c_{ij}, resulting from assigning locomotive j to route i. If the resulting route is infeasible, cost c_{ij} is set equal to infinity, i.e., $c_{ij} = +\infty$. Given the resulting cost matrix $[c_{ij}]$, we then solve the corresponding assignment problem using the Hungarian algorithm to compute the optimal assignment of the locomotives to the routes.

5. Computational experiments

Computational experiments were conducted to test the effectiveness and efficiency of the proposed models and algorithms, using a 64-bit Windows 10 system with an Intel Core i7 3.40-GHz CPU and 16-GB RAM. All the algorithms were implemented using the C++ programming language, and the IPs were solved using CPLEX 12.10, with the default parameter settings.

5.1. Experimental data

The main characteristics of the production environment and actual transportation data used are summarized as follows. The steel company that we studied has three BFs and two RSs. The four tapholes of each BF are tapped in turn. Each RS can handle no more than two loaded TPCs at a time. The different BFs and RSs are connected via tracks which are used by locomotives to tow the TPCs to their specified destinations. Here, seven locomotives are available to transport TPCs. In practice, the routing decision for locomotives is shift dependent. Generally, the three BFs tap the molten iron approximately 30 times per shift. For each tapping, four transportation requests are generated: empty TPC from BA to BF, loaded TPC from BF to BA, and from BA to RS, and empty TPC from RS to BA. Thus, the locomotives serve approximately 120 transportation requests per shift. The travel time between the BA and RS is approximately 10 minutes and that between the BA and the three BFs is 3 to 20 minutes. For each transportation request, the service time at the pickup and delivery locations is 2 minutes, and the time window at the pickup or delivery location ranges from 15 to 20 minutes.

According to the actual data features mentioned above and the mathematical formulations, the main factors affecting the complexity of the problem include the length of the scheduling horizon and the number of transportation requests and locomotives. The data from the steel company were collected across 21 shifts, over a period of one week. The number of requests during each shift ranges from 109 to 142. To further test the performance robustness of the proposed ALNS algorithm, we generated random test datasets with more diverse parameter configurations than those of the case study. For consistency with the actual scheduling, the number of locomotives was set equal to seven for all the datasets. For one shift, we generated three datasets LRP-Ks, with the number of requests K equal to 100, 120, and 150. Similarly, three datasets LRP-Ks (K = 200, 240, and 300) and two datasets LRP-Ks (K = 360 and 450) were generated for two and three shifts accordingly. To evaluate the extent to which the ALNS solution differs from the optimal one, five relatively small datasets LRP-Ks (K = 20, 30, 40, 50, and 80) were also generated, some of which could be optimally solved by CPLEX. For each dataset, 10 instances were randomly generated based on the actual data. For the tuning of parameters, 11 instances indexed by LRP_K# were additionally generated, where K (20-450) is the number of requests.

5.2. Parameters tuning

There are 15 main user-controlled parameters of the ALNS algorithm, classified into three classes (Demir et al., 2012; Sun et al., 2020b), as summarized in Table 1. The parameters in class I control the selection of operators, whereas the class II parameters are related to the SA algorithm. Class III contains the remaining parameters used in the removal and insertion operators. The parameter c_e in Class IV is the locomotive capacity when transporting empty requests together, while c_l is the capacity for transporting loaded requests together. The parameter-tuning method introduced by Ropke & Pisinger (2006a) was adopted in this study. It modifies the parameter classes sequentially and individually to inspect their impact on the quality of solutions, while maintaining the setting affording the best trade-off between solution quality and computational time. The ALNS algorithm was run ten times without the local search procedure for each instance with one specific parameter combination. Note that the final choice in the parameter values here may not be the optimal ones but can result in a good overall performance in these test runs.

The control parameters, π_1, π_2, and π_3, of the first parameter class used in the roulette wheel mechanism were tuned according to the settings introduced by Sun et al. (2020b). There are seven value combinations of π_1, π_2, and π_3. Here $(1, 5, 3)$ was the best option, among the different combinations. This setting contradicts the expectation that $\pi_1 \geq \pi_2 \geq \pi_3$, which rewards the finding of the new best solution the most. However, it can diversify the search of the ALNS algorithm. Fig.8 shows the behavior of the ALNS algorithm (with

Table 1
Values of parameters.

Class	*Notation*	*Description*	*Tuned value*
I	NI	Total number of iterations	25000
	NIS	Number of iterations in a segment	100
	π_1	Score of finding a new best solution	1
	π_2	Score of finding a new solution better than the current solution	5
	π_3	Score of accepting a new solution worse than the current solution	3
	ρ	Roulette wheel mechanism parameter	0.1
II	T^{org}	Initial temperature	100
	cr	Cooling rate	0.999
III	ϕ_1	First Shaw parameter	4
	ϕ_2	Second Shaw parameter	4
	ϕ_3	Third Shaw parameter	1
	ϕ_4	Fourth Shaw parameter	1
	ϕ_5	Fifth Shaw parameter	3
	μ	Destroy rate for removal operators	0.1
	ϵ	noise parameter	0.025
IV	c_e	Locomotive capacity for transporting empty requests	4
	c_l	Locomotive capacity for transporting loaded requests	2

local search procedure embedded) when solving LRP_200_01. On this instance, the objective value of the best known solution (BKS) computed by the algorithm over 10 runs is equal to 5694. The variations of the best, current, and new solutions over 25000 iterations are depicted in this figure.

Interestingly, for small-scale instances with 20 requests, the ALNS algorithm does not remain stable when determining the optimal solution under the setting of μ (=0.1), However, it can determine the optimal solution in each run when μ is increased. But for large-scale instances, e.g., instances with 450 requests, the solution quality decreases along with an increment in μ. This confirms that it is not necessary to remove a larger number of requests during the destroy solution phase (Ropke & Pisinger, 2006a). Detailed parameter-tuning results are presented in Table 5 to Table 9, in the Online Supplement.

5.3. Performance of operators and the local search procedure

The detailed results of the ALNS algorithm and the local search procedure under various settings are shown in Table 10 to Table 13, in the Online Supplement. Table 10 reports the performance of all the removal operators during a single run of the ALNS algorithm for all test instances in LRP_K#. For each type of settings in Table 11 to Table 13, the ALNS algorithm or the local search procedure were run ten times for all the test instances in LRP_K#.

There are five rows in Table 10, i.e., $Single$, Sum, $Number$, $Best$, $Better$, and $Worse$, denoting the time taken for a single run, the total running time, the number of times the operator is selected, the number of times the best solution is found, the number of times a better solution is found, and the number of times a worse solution is found and accepted. Statistically, $EDOR$ and RR perform the best in improving the best solution and the current solution, respectively. As for the insertion operators, RSI exhibits the worst performance. By contrast, the RI operators perform well but are time consuming. The operator FFI also performs well in improving the best solution according to the computation time, as compared with RI_2.

To evaluate the performance of the removal and insertion operators, a setting was applied to the ALNS algorithm, whereby one operator was disabled each time. As shown in Table 11, the ALNS algorithm without an adaptive mechanism (LNS) is the fastest but featured the worst solution quality. The ALNS algorithm with full machinery (FM) shows the opposite result. Evidently, the SR has the largest influence

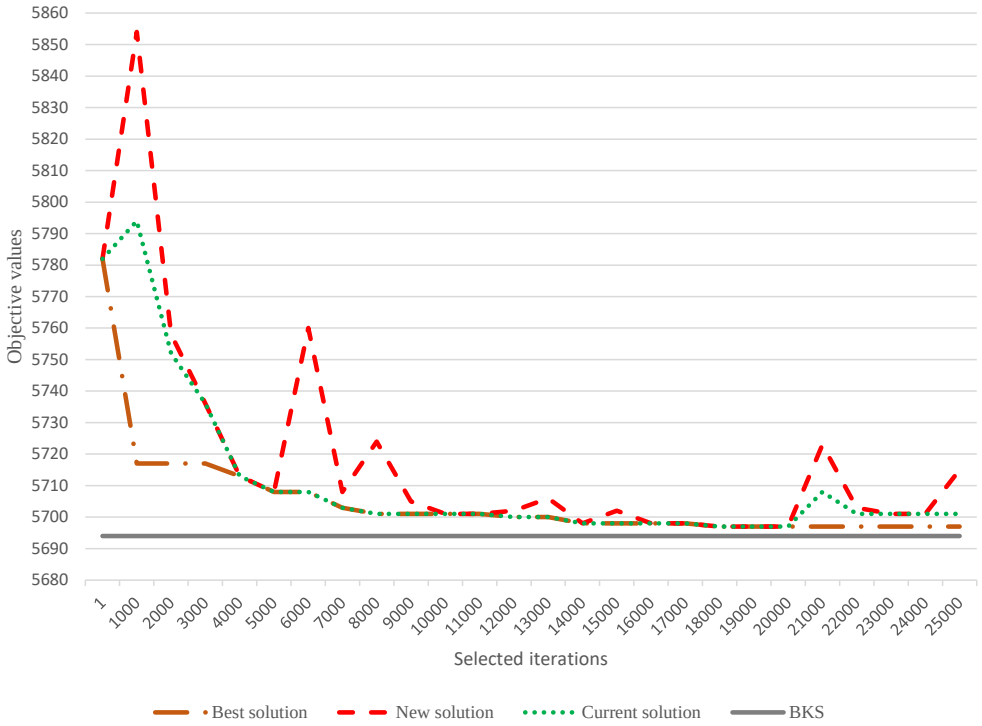

Fig. 8. Objective values found by the ALNS algorithm for a 200-request instance.

on the solution quality, as compared with the other operators. Moreover, the operators proposed for *block re-optimization*, from TOR to COR, also performs well. With regard to the insertion operators, RI_2 exhibits largest influence on solution quality.

The performance of the local search procedure under various settings is reported in Table 12. For each computation, a neighborhood structure is forbidden. Clearly, the $ISRR$ has the largest influence on solution quality, whereas IRR has the smallest influence on calculation time. Additionally, the local search procedure with full machinery (column FM) performs the best in improving the objective value.

The performance of the local search procedure when using different neighbor acceptance criteria is reported in Table 13. It describes the relative computational results of the ALNS algorithm without a local search procedure, with RVND-I, with RVND-II, and with the local search procedure embedded, i.e., $NONE$, LS_I, LS_{II}, and LS_{new}, respectively. The following results are reported: the best objective value ($Best$), average objective value ($Aver$), average running time ($Time$), and deviations ($D_B(\%)$ and $D_A(\%)$) compared to the best-known-solution (BKS).

The ALNS algorithm without a local search procedure has the shortest computation time but the worst solution quality, with $D_B(\%)$ and $D_A(\%)$ of 0.16% and 0.38%, respectively. The ALNS algorithm with the local search procedure embedded clearly performs the best. The average increment of calculation time is approximately 10 s, whereas the aforementioned two deviations decrease to 0.01% and 0.09%, respectively. Thus, it can be concluded that a good trade-off between solution quality and computation time is achieved by the local search procedure.

Table 2
Results on actual data of 21 shifts.

Day	$\|\mathcal{R}\|$	$\|\mathcal{R}_L\|$	$\|\mathcal{R}_E\|$	*OV*			*WTMI*			*TTL*			*Time*
				$CPLEX_{LB}$	$ALNS_A$	$Gap(\%)$	MM	$ALNS_A$	$Imp(\%)$	MM	$ALNS_A$	$Imp(\%)$	
1st	128	62	66	2810.00	3171.8	11.41	2451	2334.0	4.77	1439	837.8	41.78	21.71
	134	67	67	3078.59	3307.2	6.91	2553	2437.8	4.51	1567	869.4	44.52	22.47
	109	56	53	2353.47	2640.9	10.88	2001	1914.0	4.35	1206	726.9	39.73	14.60
2nd	142	70	72	3200.45	3489.9	8.29	2692	2578.0	4.23	1605	911.9	43.18	28.04
	113	56	57	2737.12	2990.4	8.47	2374	2252.0	5.14	1370	738.4	46.10	17.00
	118	59	59	2801.75	3028.7	7.49	2382	2299.0	3.48	1369	729.7	46.70	17.09
3rd	112	56	56	2282.97	2530.8	9.79	1888	1789.0	5.24	1237	741.8	40.03	17.26
	135	67	68	2936.15	3218.5	8.77	2459	2331.0	5.21	1519	887.5	41.57	24.91
	126	64	62	3419.61	3561.0	3.97	2787	2637.0	5.38	1563	924.0	40.88	22.07
4th	123	61	62	2844.77	3088.4	7.89	2438	2326.4	4.58	1378	762.0	44.70	19.96
	130	65	65	2870.30	3144.9	8.73	2387	2291.0	4.02	1551	853.9	44.95	25.35
	121	62	59	2740.84	2958.1	7.34	2284	2170.0	4.99	1315	788.1	40.07	20.82
5th	114	57	57	2466.82	2693.6	8.42	1968	1958.0	0.51	1327	735.6	44.57	16.04
	128	64	64	3492.45	3571.0	2.20	2805	2691.0	4.06	1548	880.0	43.15	23.86
	141	70	71	3580.59	3839.1	6.73	2846	2812.0	1.19	1891	1027.1	45.68	29.50
6th	129	66	63	3153.22	3265.4	3.44	2471	2335.8	5.47	1609	929.6	42.22	24.04
	131	65	66	3801.94	3985.6	4.61	3676	3100.0	15.67	1508	885.6	41.27	25.38
	124	62	62	2535.31	2843.2	10.83	2142	2019.0	5.74	1427	824.2	42.24	22.36
7th	123	61	62	2836.97	3062.5	7.36	2352	2242.0	4.68	1382	820.5	40.63	19.41
	121	60	61	3325.76	3431.7	3.09	2715	2585.0	4.79	1504	846.7	43.70	21.31
	120	58	62	2491.72	2801.8	11.07	2105	2009.0	4.56	1401	792.8	43.41	20.09
Average	125	62	63	2940.99	3172.6	7.30	2466	2338.6	4.89	1463	834.0	42.91	21.58

5.4. Computational results for actual instances

To verify the effectiveness of the proposed ALNS algorithm in practice, we conducted a set of experiments based on the actual datasets collected from a large steel company in China. The results are compared with the rule-based manual scheduling method (denoted as MM), currently used by the steel company that motivated our study. This method is a type of greedy procedure. First, it sorts all the transportation requests in a non-increasing order, according to the earliest pickup time at the pickup locations. Then, it allocates the transportation requests sequentially to the earliest available locomotive. Table 2 shows the computational results of the actual datasets, including the problem structure (the number of total requests $|\mathcal{R}|$, the number of loaded TPC requests $|\mathcal{R}_L|$, and the number of empty TPC requests $|\mathcal{R}_E|$), objective values OV, molten iron waiting time of loaded TPCs ($WTMI$), and travel time of locomotives (TTL). The lower bounds found by CPLEX based on three-index formulation are reported in the column $CPLEX_{LB}$. It also presents the solutions obtained via the ALNS algorithm (the average out of ten runs, i.e., $ALNS_A$). It indicates the optimal gap ($Gap(\%)$) between the ALNS solution and the lower bound found by CPLEX. And the improvements ($Imp(\%)$) in waiting time of molten iron and travel time of locomotives when using the ALNS algorithm, as compared to the manual method. Additionally, it reports the computation time of the ALNS algorithm.

As shown in Table 2, the average of the optimal gap is 7.30%. Even though the running time of CPLEX was set to 24 hours, no feasible solution was obtained based on the three-index formulation. Besides that, CPLEX cannot find any lower bound or feasible solution when using the group-based formulation. The ALNS algorithm can, on average, improve the waiting time of molten iron in loaded TPCs by 4.89% and the locomotive travel time by 42.91%. The decreased waiting time of the molten iron in loaded TPCs implies that the temperature loss is reduced, which, in turn, reduces the energy consumption. The reduction of locomotive travel time also implies an improvement in the locomotive operation efficiency. Moreover, the computation time of the ALNS algorithm is less than 30 s, for all the instances; by contrast, the manual method typically requires over an hour to generate a feasible solution. This implies that the proposed algorithm is advantageous in terms of work efficiency.

5.5. Computational results for extensive instances

To evaluate the deviation between the ALNS solution and the optimal one, the ALNS algorithm and CPLEX were tested on instances with sizes ranging from 20 to 100. We have mentioned earlier that the planner of a steel company takes more than an hour to make a transportation plan for one shift. Therefore, the upper bound 7200s was chosen as the running time limit of CPLEX. Additionally, the ALNS algorithm was tested on large instances, as a performance evaluation. All the instances were solved ten times using the ALNS algorithm. CPLEX cannot compute the lower bounds, based on the three-index and group-based formulations, for several instances involving more than 100 requests. We, therefore, report the lower bounds values only for instances involving up to 100 requests.

Table 3
Computational results for instances containing 20-50 requests.

Inst	*MM*	$CPLEX_I$			$CPLEX_{II}$			$ALNS_f$			*Imp*(%)		*Gap*(%)
		UB	*LB*	*Time*	*UB*	*LB*	*Time*	*Best*	*Aver*	*Time*	Imp_M	Imp_C	
LRP_20_01	678	605	605	453.30	605	605	2112.86	605	605	1.45	10.77	*	*
LRP_20_02	868	836	836	227.23	836	836	224.39	836	836	1.56	3.69	*	*
LRP_20_03	649	579	579	4245.75	579	579	1288.50	579	579	1.33	10.79	*	*
LRP_20_04	732	645	645	5426.22	645	645	120.13	645	645	1.37	11.89	*	*
LRP_20_05	657	598	598	2911.14	598	598	3711.64	598	598	1.34	8.98	*	*
LRP_20_06	609	581	581	586.41	581	581	3614.59	581	581	1.39	4.60	*	*
LRP_20_07	633	574	574	331.45	574	574	1322.08	574	574	1.35	9.32	*	*
LRP_20_08	936	866	866	197.14	866	866	209.09	866	866	1.33	7.48	*	*
LRP_20_09	802	709	709	265.78	709	709	1422.59	709	709	1.54	11.60	*	*
LRP_20_10	615	528	528	514.13	528	528	2068.73	528	528	1.31	14.15	*	*
Average	717.9	652.1	652.1	1515.86	652.1	652.1	1609.46	652.1	652.1	1.40	9.32	*	*
LRP_30_01	1054	942	932.38	7200*	941	937.23	7200*	941	941	1.85	10.72	*	0.40
LRP_30_02	1176	1011	1003.00	7200*	1011	1011.00	3289.55	1011	1011	1.94	14.03	*	*
LRP_30_03	1037	864	859.00	7200*	888	860.00	7200*	864	864	1.92	16.68	*	0.46
LRP_30_04	1017	919	904.00	7200*	919	897.00	7200*	919	919	1.93	9.64	*	1.63
LRP_30_05	1154	955	955.00	5972.34	955	955.00	4883.25	955	955	1.80	17.24	*	*
LRP_30_06	1107	946	938.50	7200*	946	946.00	6294.23	946	946	2.05	14.54	*	*
LRP_30_07	1089	971	971.00	692.45	971	971.00	699.28	971	971	1.78	10.84	*	*
LRP_30_08	1318	1204	1204.00	5304.55	1204	1204.00	717.05	1204	1204	2.17	8.65	*	*
LRP_30_09	988	826	826.00	1400.66	826	826.00	1288.95	826	826	1.87	16.40	*	*
LRP_30_10	981	881	876.00	7200*	887	875.07	7200*	881	881	2.09	10.19	*	0.57
Average	1092.1	951.9	946.89	5657	954.8	948.23	4597.23	951.8	951.8	1.94	12.89	*	0.29
LRP_40_01	1206	1071	1071.00	4975.14	1071	1070.92	1935.17	1071	1071	3.45	11.19	*	*
LRP_40_02	1402	1188	1081.01	7200*	—	1156.50	7200*	1176	1176	3.10	16.12	1.01	1.66
LRP_40_03	1270	1050	1035.00	7200*	1084	1041.00	7200*	1050	1050	3.38	17.32	*	0.86
LRP_40_04	1526	1312	1300.50	7200*	1312	1290.50	7200*	1312	1312	2.97	14.02	*	0.88
LRP_40_05	1318	1171	1169.00	7200*	1171	1171.00	7119.02	1171	1171	3.05	11.15	*	*
LRP_40_06	1333	1117	1105.56	7200*	—	1110.00	7200*	1117	1117	3.19	16.20	*	0.63
LRP_40_07	1273	1057	1051.00	7200*	—	1036.67	7200*	1057	1057	3.09	16.97	*	0.57
LRP_40_08	1320	1226	1222.00	7200*	1226	1225.89	2737.41	1226	1226	3.30	7.12	*	*
LRP_40_09	1544	1259	1254.40	7200*	1259	1242.00	7200*	1259	1259	3.54	18.46	*	0.37
LRP_40_10	1184	1025	1008.15	7200*	1023	1020.00	7200*	1023	1023	3.19	13.60	*	0.29
Average	1337.6	1147.6	1129.76	6977.51	1163.71	1136.15	6219.16	1146.2	1146.2	3.23	14.22	0.10	0.53
LRP_50_01	1706	1460	1413.17	7200*	—	1432.50	7200*	1458	1458.1	4.21	14.53	0.13	1.76
LRP_50_02	1718	1486	1367.24	7200*	—	1421.00	7200*	1434	1435.9	3.89	16.42	3.37	1.04
LRP_50_03	1812	—	1399.36	7200*	1477	1440.50	7200*	1454	1454.4	4.55	19.74	1.53	0.96
LRP_50_04	1655	1328	1257.88	7200*	1320	1291.21	7200*	1305	1305.6	4.36	21.11	1.09	1.10
LRP_50_05	1547	1338	1288.59	7200*	—	1308.50	7200*	1331	1331.5	4.49	13.93	0.49	1.73
LRP_50_06	1695	1405	1325.69	7200*	—	1319.00	7200*	1388	1388.0	4.53	18.11	1.21	4.49
LRP_50_07	1541	1343	1201.66	7200*	—	—	7200*	1285	1285.2	3.90	16.60	4.30	6.50
LRP_50_08	1569	—	1233.41	7200*	1310	1292.00	7200*	1302	1302.5	3.81	16.99	0.57	0.81
LRP_50_09	1704	1475	1438.08	7200*	1465	1446.00	7200*	1465	1465.0	4.03	14.03	0.00	1.30
LRP_50_10	1649	1310	1262.25	7200*	—	1294.33	7200*	1303	1303.7	4.32	20.94	0.48	0.72
Average	1659.6	1393.1	1318.73	7200*	1393.0	1360.56	7200*	1372.5	1373.0	4.21	17.24	1.32	2.01

Table 3 and Table 4 report the comparison between the proposed algorithm ($ALNS_f$) and CPLEX in solving instances with 20-50 and 80-100 requests, respectively. Columns Imp_M and Imp_C in Table 3 report the percentage improvements of the ALNS solution compared to the manual method (MM) and the best upper bound (UB) of CPLEX, which are respectively calculated as $100(MM - Aver)/MM$ and $100(UB - Aver)/UB$. To further investigate the solution quality of the proposed algorithm, the optimal gaps of the ALNS solution compared to the lower bounds (LB) computed by CPLEX are presented in column

$Gap(\%)$, calculated as $100(Best-LB)/Best$. Since CPLEX cannot determine any upper bound for instances with sizes larger than 50, only the optimal gaps are reported in Table 4. These bounds found via CPLEX are based on the three-index ($CPLEX_I$) and group-based formulation ($CPLEX_{II}$). Note that the lower or upper bound used for comparison is the better one of bounds found by CPLEX, based on those two formulations.

The results presented in Table 3 indicate that both the ALNS algorithm and CPLEX can optimally solve instances with 20 requests. However, the average computation time for the ALNS algorithm, i.e., 1.4 s, is negligible when compared to that for CPLEX. CPLEX cannot solve to optimality all instances of groups LRP_30 and LRP_40, and for those instances that can be optimally solved by CPLEX, the ALNS algorithm can also find corresponding optimal solutions. Significantly, deviations between the best and average objective values of instances with sizes less than 50 are equal. Moreover, there is no guarantee that CPLEX can determine a feasible solution within 7200 s for instances with 50 requests. Further, the solution quality of the ALNS algorithm is better than that of CPLEX, based on the percentage improvements of 1.32%. As shown in Table 3 and Table 4, compared to the manual method, with the increment of instance scale, the improvement upon solution quality of the ALNS algorithm gradually raises and tends to be stable.

Table 4
Computational results for instances containing 80 and 100 requests.

Inst	*MM*	$CPLEX_I$			$CPLEX_{II}$			$ALNS_f$			*Gap*(%)	Imp_M
		UB	*LB*	*Time*	*UB*	*LB*	*Time*	*Best*	*Aver*	*Time*		
LRP_80_01	2947	—	2475.49	7200*	—	—	7200*	2551	2553.2	9.98	3.04	13.44
LRP_80_02	2962	—	2300.87	7200*	—	—	7200*	2439	2440.2	10.47	5.71	17.66
LRP_80_03	2610	—	1915.97	7200*	—	—	7200*	2122	2124.3	9.48	9.81	18.70
LRP_80_04	2850	—	2281.60	7200*	—	2347.0	7200*	2398	2399.1	9.71	2.17	15.86
LRP_80_05	2775	—	2167.97	7200*	—	—	7200*	2323	2324.0	10.60	6.71	16.29
LRP_80_06	2550	—	1901.70	7200*	—	2077.5	7200*	2090	2090.3	10.03	0.61	18.04
LRP_80_07	2705	—	1969.30	7200*	—	—	7200*	2177	2178.6	9.32	9.61	19.52
LRP_80_08	2818	—	2110.86	7200*	—	—	7200*	2287	2288.6	10.09	7.77	18.84
LRP_80_09	2699	—	2147.26	7200*	—	—	7200*	2328	2329.3	9.91	7.82	13.75
LRP_80_10	2800	—	2349.75	7200*	—	—	7200*	2458	2458.1	9.47	4.41	12.21
Average	2771.6	—	2162.08	7200*	—	2212.3	7200*	2317.3	2318.6	9.91	5.77	16.43
LRP_100_01	3566	—	2520.57	7200*	—	—	7200*	2829	2831.6	16.44	10.98	20.67
LRP_100_02	3602	—	2623.43	7200*	—	—	7200*	2884	2884.8	15.36	9.06	19.93
LRP_100_03	3627	—	2658.36	7200*	—	—	7200*	2973	2973.2	15.81	10.59	18.03
LRP_100_04	3362	—	2571.30	7200*	—	2824	7200*	2838	2838.0	15.48	0.49	15.59
LRP_100_05	3393	—	2514.79	7200*	—	—	7200*	2796	2800.6	13.61	10.21	17.60
LRP_100_06	3624	—	2675.63	7200*	—	—	7200*	2973	2974.8	15.92	10.06	17.96
LRP_100_07	3351	—	2455.83	7200*	—	—	7200*	2665	2666.7	13.60	7.91	20.47
LRP_100_08	3499	—	2695.33	7200*	—	—	7200*	2977	2978.8	15.56	9.52	14.92
LRP_100_09	3238	—	2482.97	7200*	—	—	7200*	2704	2710.1	13.49	8.38	16.49
LRP_100_10	3806	—	2785.31	7200*	—	—	7200*	3081	3083.8	15.65	9.68	19.05
Average	3506.8	—	2598.35	7200*	—	2824	7200*	2872	2874.2	15.09	8.69	18.07

Table 3 and Table 4 show that the lower bound of the group-based formulation, thanks to the definition of its decision variables, is generally better than the one from the three-index formulation, especially for instances with requests more than 40 requests. However, due to the larger size of the group-based formulation, CPLEX cannot always compute lower bounds for instances with more than 40 requests. Table 3 shows that the ALNS solved to optimality several instances involving up to 40 requests and that the maximum average optimality gap of the ALNS is equal to about 2.0 per cent, hereby certifying its effectiveness.

The results of Table 4 shows that more significant optimality gaps are computed on instances involving 80 and 100 requests, with the maximum average optimality gap equal to 8.7 per cent. This is probably due to the deterioration of the lower bound computed by CPLEX on larger instances, for which CPLEX reached the imposed time limit for all the instances.

To further evaluate the stability and performance of the ALNS algorithm, we tested instances with sizes

ranging from 120 to 450. Detailed computational results of the ALNS algorithm, including instances with sizes ranging from 50 to 100, are reported in Table 14 of the Online Supplement. Deviations between the best and average objective values vary from 0-0.23% as shown in column $D(\%)$. Moreover, the value of 0.06%, i.e., the average of the deviations, is significantly small and thus verifies the stability of the proposed algorithm.

6. Conclusions

Herein, we studied a new locomotive routing problem in which a homogeneous fleet of locomotives with different start and end locations is available, and a set of heterogeneous requests must be routed subject to time windows, capacity, incompatibility, and LIFO constraints. Both three-index and group-based formulations were established for this problem. An ALNS algorithm that includes adapted and novel operators was designed to generate high-quality solutions. Besides, a local search procedure equipped with adapted neighborhood structures and greedy acceptance criterion was designed to further improve the solution quality of the ALNS algorithm.

The comparison results show that the proposed algorithm significantly outperforms CPLEX in both computational time and solution quality. The ALNS algorithm takes less than one percent of the time to obtain higher quality solutions compared with the current method used by steel companies. The stability of our algorithm was verified by the small deviations between the best and average objective values, for all the test instances.

It should be noted that real-world locomotive routing poses several challenging extensions to this problem. The steel company mentioned herein does not involve a pretreatment process. However, in other companies, a pretreatment referred to as desulfurization may be conducted before unloading the molten iron. Integrating this additional constraint would render the studied LRP as a more challenging one. Our future work will, therefore, be directed toward considering these extensions and the other important features associated with this class of problems.

Acknowledgments

This work was supported by the Major Program of National Natural Science Foundation of China (72192830, 72192835), the National Natural Science Foundation of China (72102034, 71672032), and the 111 Project (B16009). The authors are also grateful to associate editor and three anonymous referees for their meticulous reviews, which have greatly improved the content and presentation of this paper.

References

Alyasiry, A. M., Forbes, M., & Bulmer, M. (2019). An Exact Algorithm for the Pickup and Delivery Problem with Time Windows and Last-in-First-out Loading. *Transportation Science*, *53*, 1695–1705. doi:10.1287/trsc.2019.0905.

Baldacci, R., Battarra, M., & Vigo, D. (2008). Routing a Heterogeneous Fleet of Vehicles. In B. Golden, S. Raghavan, & E. Wasil (Eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges* (pp. 3–27). Boston, MA: Springer US volume 43. URL: http://link.springer.com/10.1007/978-0-387-77778-8_1. doi:10.1007/978-0-387-77778-8_1 iSSN: 1387-666X Series Title: Operations Research/Computer Science Interfaces.

Baldacci, R., & Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming*, *120*, 347–380. doi:10.1007/s10107-008-0218-9.

Baldacci, R., Mingozzi, A., & Roberti, R. (2012). Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, *218*, 1–6. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221711006692. doi:10.1016/j.ejor.2011.07.037.

Battarra, M., Erdoğan, G., Laporte, G., & Vigo, D. (2010). The Traveling Salesman Problem with Pickups, Deliveries, and Handling Costs. *Transportation Science*, *44*, 383–399. doi:10.1287/trsc.1100.0316.

Benavent, E., Landete, M., Mota, E., & Tirado, G. (2015). The multiple vehicle pickup and delivery problem with LIFO constraints. *European Journal of Operational Research*, *243*, 752–762. doi:10.1016/j.ejor.2014.12.029.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *TOP*, *15*, 1–31. doi:10.1007/s11750-007-0009-0.

Cheang, B., Gao, X., Lim, A., Qin, H., & Zhu, W. (2012). Multiple pickup and delivery traveling salesman problem with last-in-first-out loading and distance constraints. *European Journal of Operational Research*, *223*, 60–75. doi:10.1016/j.ejor.2012.06.019.

Chen, C., Demir, E., & Huang, Y. (2021). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *European Journal of Operational Research*, *294*, 1164–1180. doi:10.1016/j.ejor.2021.02.027.

Cherkesly, M., Desaulniers, G., Irnich, S., & Laporte, G. (2016). Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks. *European Journal of Operational Research*, *250*, 782–793. doi:10.1016/j.ejor.2015.10.046.

Cherkesly, M., Desaulniers, G., & Laporte, G. (2015a). Branch-Price-and-Cut Algorithms for the Pickup and Delivery Problem with Time Windows and Last-in-First-Out Loading. *Transportation Science*, *49*, 752–766. doi:10.1287/trsc.2014.0535.

Cherkesly, M., Desaulniers, G., & Laporte, G. (2015b). A population-based metaheuristic for the pickup and delivery problem with time windows and LIFO loading. *Computers & Operations Research*, *62*, 23–35. doi:10.1016/j.cor.2015.04.002.

Cherkesly, M., & Gschwind, T. (2022). The pickup and delivery problem with time windows, multiple stacks, and handling operations. *European Journal of Operational Research*, *301(2)*, 647–666. doi:10.1016/j.ejor.2021.11.021.

Cordeau, J.-F., Iori, M., Laporte, G., & Salazar González, J. J. (2010). A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. *Networks*, *55*, 46–59. doi:10.1002/net.20312.

Costa, L., Contardo, C., & Desaulniers, G. (2019). Exact Branch-Price-and-Cut Algorithms for Vehicle Routing. *Transportation Science*, *53*, 946–985. doi:10.1287/trsc.2018.0878.

Demir, E., Bektaş, T., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the Pollution-Routing Problem. *European Journal of Operational Research*, *223*, 346–359. doi:10.1016/j.ejor.2012.06.044.

Deng, M., Inoue, A., & Kawakami, S. (2011). Optimal path planning for material and products transfer in steel works using aco. In *The 2011 International Conference on Advanced Mechatronic Systems* (pp. 47–50). Zhengzhou, China: IEEE.

Franceschetti, A., Demir, E., Honhon, D., Van Woensel, T., Laporte, G., & Stobbe, M. (2016). A metaheuristic for the time-dependent pollution-routing problem. *European Journal of Operational Research*, *259(3)*, 972–991.

Geiger, M. J., Kletzander, L., & Musliu, N. (2019). Solving the Torpedo Scheduling Problem. *Journal of Artificial Intelligence Research*, *66*, 1–32. doi:10.1613/jair.1.11303.

Gendreau, M., & Potvin, J.-Y. (Eds.) (2019). *Handbook of Metaheuristics* volume 272 of *International Series in Operations Research & Management Science*. Cham: Springer International Publishing. doi:10.1007/978-3-319-91086-4.

Goldwaser, A., & Schutt, A. (2018). Optimal Torpedo Scheduling. *Journal of Artificial Intelligence Research*, *63*, 955–986. doi:10.1613/jair.1.11268.

Harbaoui Dridi, I., Ben Alaïa, E., Borne, P., & Bouchriha, H. (2020). Optimisation of the multi-depots pick-up and delivery problems with time windows and multi-vehicles using PSO algorithm. *International Journal of Production Research*, *58*, 4201–4214. doi:10.1080/00207543.2019.1650975.

Hornstra, R. P., Silva, A., Roodbergen, K. J., & Coelho, L. C. (2020). The vehicle routing problem with simultaneous pickup and delivery and handling costs. *Computers & Operations Research*, *115*, 104858. doi:10.1016/j.cor.2019.104858.

Huang, H., Chai, T., Luo, X., Zheng, B., & Wang, H. (2011). Two-Stage Method and Application for Molten Iron Scheduling Problem between Iron-Making Plants and Steel-Making Plants. *IFAC Proceedings Volumes*, *44*, 9476–9481. doi:10.3182/20110828-6-IT-1002.01373.

Kikuchi, J., Konishi, M., & Imai, J. (2008). Transfer planning of molten metals in steel worksby decentralized agent. *Memoirs of the Faculty of Engineering, Okayama University*, *42*, 60–70.

Kletzander, L., & Musliu, N. (2017). A Multi-stage Simulated Annealing Algorithm for the Torpedo Scheduling Problem. In D. Salvagnin, & M. Lombardi (Eds.), *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (pp. 344–358). Padua, Italy: Springer, Cham. doi:10.1007/978-3-319-59776-8_28.

Li, J.-q., Pan, Q.-k., & Duan, P.-y. (2016a). An Improved Artificial Bee Colony Algorithm for Solving Hybrid Flexible Flowshop With Dynamic Operation Skipping. *IEEE Transactions on Cybernetics*, *46*, 1311–1324. doi:10.1109/TCYB.2015.2444383.

Li, Y., Chen, H., & Prins, C. (2016b). Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. *European Journal of Operational Research*, *252*, 27–38. doi:doi:10.1016/j.ejor.2015.12.032.

Li, Y., Lim, A., Oon, W.-C., Qin, H., & Tu, D. (2011). The tree representation for the pickup and delivery traveling salesman

problem with LIFO loading. *European Journal of Operational Research*, *212*, 482–496. doi:10.1016/j.ejor.2011.02.008.
Liu, F., Gui, M., Yi, C., & Lan, Y. (2019). A Fast Decomposition and Reconstruction Framework for the Pickup and Delivery Problem With Time Windows and LIFO Loading. *IEEE Access*, *7*, 71813–71826. doi:10.1109/ACCESS.2019.2920444.
Liu, Y., & Wang, G. (2015). The Mix Integer Programming Model for Torpedo Car Scheduling in Iron and Steel Industry:. Bangkok, Thailand. doi:10.2991/cisia-15.2015.199.
Lübbecke, M. E., & Zimmermann, U. T. (2003). Engine Routing and Scheduling at Industrial In-Plant Railroads. *Transportation Science*, *37*, 183–197. doi:10.1287/trsc.37.2.183.15251.
Moradi Afrapoli, A., & Askari-Nasab, H. (2019). Mining fleet management systems: a review of models and algorithms. *International Journal of Mining, Reclamation and Environment*, *33*, 42–60. doi:10.1080/17480930.2017.1336607.
Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2008). A survey on pickup and delivery problems: Part I: Transportation between customers and depot. *Journal für Betriebswirtschaft*, *58*, 21–51. doi:10.1007/s11301-008-0033-7.
Peng, B., Zhang, Y., Lü, Z., Cheng, T., & Glover, F. (2020). A learning-based memetic algorithm for the multiple vehicle pickup and delivery problem with LIFO loading. *Computers & Industrial Engineering*, *142*, 106241. doi:10.1016/j.cie.2019.106241.
Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, *34*, 2403–2435. doi:10.1016/j.cor.2005.09.012.
Piu, F., Prem Kumar, V., Bierlaire, M., & Speranza, M. (2015). Introducing a preliminary consists selection in the locomotive assignment problem. *Transportation Research Part E: Logistics and Transportation Review*, *82*, 217–237. doi:10.1016/j.tre.2015.07.003.
Pollaris, H., Braekers, K., Caris, A., Janssens, G. K., & Limbourg, S. (2015). Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum*, *37*, 297–330. doi:10.1007/s00291-014-0386-3.
Potvin, J.-Y., & Rousseau, J.-M. (1995). An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, *46*, 1433–1446. doi:https://doi.org/10.1057/jors.1995.204.
Ropke, S., & Pisinger, D. (2006a). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, *40*, 455–472. doi:10.1287/trsc.1050.0135.
Ropke, S., & Pisinger, D. (2006b). A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, *171*, 750–775. doi:10.1016/j.ejor.2004.09.004.
Souza, M., Coelho, I., Ribas, S., Santos, H., & Merschmann, L. (2010). A hybrid heuristic algorithm for the open-pit-mining operational planning problem. *European Journal of Operational Research*, *207*, 1041–1051. doi:10.1016/j.ejor.2010.05.031.
Subramanian, A., Drummond, L., Bentes, C., Ochi, L., & Farias, R. (2010). A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Computers & Operations Research*, *37*, 1899–1911. doi:10.1016/j.cor.2009.10.011.
Sun, D., Meng, Y., Tang, L., Liu, J., Huang, B., & Yang, J. (2020a). Storage space allocation problem at inland bulk material stockyard. *Transportation Research Part E: Logistics and Transportation Review*, *134*, 101856. doi:10.1016/j.tre.2020.101856.
Sun, P., Veelenturf, L. P., Hewitt, M., & Van Woensel, T. (2020b). Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, *138*, 101942. doi:10.1016/j.tre.2020.101942.
Tang, L., Liu, J., Rong, A., & Yang, Z. (2001). A review of planning and scheduling systems and methods for integrated steel production. *European Journal of Operational Research*, *133*, 1–20. doi:10.1016/S0377-2217(00)00240-X.
Tang, L., & Meng, Y. (2021). Data analytics and optimization for smart industry. *Frontiers of Engineering Management*, *8*, 157–171. doi:https://doi.org/10.1007/s42524-020-0126-0.
Tang, L., Wang, G., & Liu, J. (2007). A branch-and-price algorithm to solve the molten iron allocation problem in iron and steel industry. *Computers & Operations Research*, *34*, 3001–3015. doi:10.1016/j.cor.2005.11.010.
Tang, L., Zhao, Y., & Liu, J. (2014). An Improved Differential Evolution Algorithm for Practical Dynamic Scheduling in Steelmaking-Continuous Casting Production. *IEEE Transactions on Evolutionary Computation*, *18*, 209–225. doi:10.1109/TEVC.2013.2250977.
Veenstra, M., Cherkesly, M., Desaulniers, G., & Laporte, G. (2017a). The pickup and delivery problem with time windows and handling operations. *Computers & Operations Research*, *77*, 127–140. doi:10.1016/j.cor.2016.07.014.
Veenstra, M., Roodbergen, K. J., Vis, I. F., & Coelho, L. C. (2017b). The pickup and delivery traveling salesman problem with handling costs. *European Journal of Operational Research*, *257*, 118–132. doi:10.1016/j.ejor.2016.07.009.
Wang, G., & Tang, L. (2007). A Column Generation for Locomotive Scheduling Problem in Molten Iron Transportation. In *2007 IEEE International Conference on Automation and Logistics* (pp. 2227–2233). Jinan, China: IEEE. doi:10.1109/ICAL.2007.4338946.
Wei, L., Qin, H., Zhu, W., & Wan, L. (2015). A study of perturbation operators for the pickup and delivery traveling salesman problem with LIFO or FIFO loading. *Journal of Heuristics*, *21*, 617–639. doi:10.1007/s10732-015-9293-2.

Xu, X., Li, C.-L., & Xu, Z. (2018). Integrated train timetabling and locomotive assignment. *Transportation Research Part B: Methodological*, *117*, 573–593. doi:10.1016/j.trb.2018.09.015.

Online Supplement Material

I. Group-based mathematical formulation

The group-based formulation is adapted from the pattern or group-based formulation (Lübbecke & Zimmermann (2003) and Wang & Tang (2007)). By adding to the principle of group generation, the adapted model differs from the previous ones in terms of its objective function and the operation time window constraints for requests in a group. The latter is caused by a variation of the locomotive capacity for transporting requests of different types. The new notions used in this formulation are introduced, as follows.

Let $g \in \mathcal{G}$ denote a group of requests (of one request or more) that can be transported together by a locomotive. $\mathcal{G}$ is the set of all feasible groups and includes four subsets, $\mathcal{G}_m$, where $m \in M$ and $M = \{1, 2, 3, 4\}$. The subscript m indicates the number of requests included in an element of the related group set. Evidently, if a group g contains three or four requests, then $\forall r \in g, r \in \mathcal{R}_E$. Each request $r \in \mathcal{R}$ is assigned a group set $\mathcal{G}^r$ with every element containing r. A feasible group must satisfy the condition that a locomotive exists, such that it can execute all pickup and delivery operations without violating the time window constraints. Furthermore, the capacity constraints, the incompatibility constraints, and the LIFO rule should be satisfied by each group. Each group $g \in \mathcal{G}$ is associated with a set ϑ_g^+ (ϑ_g^-), where the element in ϑ_g^+ (ϑ_g^-) is the request group or the locomotive origin (or destination), which can be a feasible predecessor (or successor) of g. Incidentally, the successor (predecessor) set of ℓ^+ (ℓ^-) is $\vartheta_{\ell^+}^-$ ($\vartheta_{\ell^-}^+$). The service time of the *ith* operation (pickup or delivery) in group g is denoted by s_{g_i}. Let $[e_{g_i}, l_{g_i}]$ be the time window of the *ith* operation in group g.

Note that the sum of the travel time between the ordered pickup and the delivery locations of group g is a constant, D_g. Hence, a route consisting of groups can be converted into a route in graph G. Therefore, the calculation of the molten iron waiting time is the same as in the three-index formulation. Considering the service start time of a group of requests (BA→RS), once the visiting time of the delivery locations are known, the total molten iron waiting time of this group can be calculated using equation (1), as described in Section 3. Here, $W(T_{g_1})$ is the molten iron waiting time of group g, where T_{g_1} is the service start time of g, which determines the visiting time of the delivery locations.

The decision variables used in the group-based formulation are as follows:

T_{g_i}: the service start time of the *ith* operation in group g;
$x_{gd}^{\ell} \in \{0, 1\}$: equals 1 if ℓ visits group g and group d successively and 0, otherwise;
$x_{\ell^+ g}^{\ell} \in \{0, 1\}$: equals 1 if g is the first group visited by ℓ and 0, otherwise;
$x_{g\ell^-}^{\ell} \in \{0, 1\}$: equals 1 if g is the last group visited by ℓ and 0, otherwise;
$W(T_{g_1})$: the total molten iron waiting time of group g with respect to its service start time T_{g_1}.

The group-based formulation is then obtained as follows.

$$min \sum_{\ell \in \mathcal{L}} \sum_{r \in \mathcal{R}_{L_1}} \sum_{g \in \mathcal{G}^r} \sum_{d \in \vartheta_g^-} x_{gd}^{\ell} W(T_{g_1}) + \sum_{\ell \in \mathcal{L}} \sum_{g \in \mathcal{G}} \sum_{d \in \vartheta_g^-} x_{gd}^{\ell} D_g + \sum_{\ell \in \mathcal{L}} \sum_{g \in \mathcal{G} \cup \ell^+} \sum_{d \in \mathcal{G} \cup \ell^-} x_{gd}^{\ell} t_{gd} \tag{18}$$

$$s.t. \sum_{g \in \mathcal{G} \cup \ell^-} x_{\ell^+ g}^{\ell} = \sum_{g \in \mathcal{G} \cup \ell^+} x_{g\ell^-}^{\ell} = 1, \forall \ell \in \mathcal{L} \tag{19}$$

$$\sum_{d \in \vartheta_g^+} x_{dg}^{\ell} - \sum_{d \in \vartheta_g^-} x_{gd}^{\ell} = 0, \forall g \in \mathcal{G}, \ell \in \mathcal{L} \tag{20}$$

$$\sum_{\ell\in\mathcal{L}}\sum_{g\in\mathcal{G}^r}\sum_{d\in\vartheta_g^-} x_{gd}^{\ell} = 1, \forall r \in \mathcal{R} \tag{21}$$

$$T_{g_i} + s_{g_i} + t_{g_i g_{i+1}} + (\sum_{d\in\vartheta_g^-} x_{gd}^{\ell} - 1)\eta \le T_{g_{i+1}}, \forall g \in \mathcal{G}_m, m \in M, i = 1, \ldots, 2m-1, \ell \in \mathcal{L} \tag{22}$$

$$T_{g_{2m}} + s_{g_{2m}} + t_{g_{2m}d_1} + \eta(x_{gd}^{\ell} - 1) \le T_{d_1}, \forall g \in \mathcal{G}_m, m \in M, d \in \{\vartheta_g^-/\ell^-\}, \ell \in \mathcal{L} \tag{23}$$

$$T_{\ell^+} + t_{\ell^+ d_1} + (x_{\ell^+ d}^{\ell} - 1)\eta \le T_{d_1}, \forall d \in \{\vartheta_{\ell^+}^-/\ell^-\}, \ell \in \mathcal{L} \tag{24}$$

$$T_{g_{2m}} + s_{g_{2m}} + t_{g_{2m}\ell^-} + (x_{g\ell^-}^{\ell} - 1)\eta \le T_{\ell^-}, \forall g \in \{\vartheta_{\ell^-}^+/\ell^+\} \cap \mathcal{G}_m, m \in M, \ell \in \mathcal{L} \tag{25}$$

$$T_{\ell^+} + t_{\ell^+\ell^-} + (x_{\ell^+\ell^-}^{\ell} - 1)\eta \le T_{\ell^-}, \ell \in \mathcal{L} \tag{26}$$

$$e_{g_i} \le T_{g_i} \le l_{g_i}, \forall g \in \mathcal{G}_m, m \in M, i = 1, \ldots, 2m \tag{27}$$

$$et_{\ell} \le T_k \le lt_{\ell}, \forall k \in \{\ell^- \cup \ell^+\}, \ell \in \mathcal{L} \tag{28}$$

The objective function (18) minimizes the sum of the molten iron waiting time and the travel time of locomotives. The locomotive travel time consists of an inner group (visited) travel time, the travel time between the origin of locomotive and the first visited group, the travel time between the groups in route, and the travel time between the last visited group and the destination of locomotive. Note that, if a locomotive ℓ has no group to serve, then the contribution of this route to the objective value, is the travel time from ℓ^+ to ℓ^-. Constraints (19) ensure that each locomotive commences at its origin and finishes at its destination. (20) are the flow conservation constraints. Constraints (21) guarantee that each request is served by only one locomotive. (22) are the constraints for computing the service start time of the operations in group. (23) denote the constraints for computing the service start time of a group succeeding another group, and (24) provide the constraints for the locomotive origin. Constraints (25) are used for computing the finishing time of a route that is not empty. Constraints (26) reflect the relationship between start and end time of an empty locomotive transfer. (27) are the time window constraints for the inner operations of a group. (28) are the time window constraints for the start time and the finish time of a route. Note that, in the first part of (18), $\mathcal{R}_{L_1}$ denotes the set of loaded TPC requests transported from BA to RS.

II. Detailed computational results

Table 5 to Table 9 show the detailed tuning results of parameters, as described in Section 5.2. Table 10 to Table 13 show the detailed performance of the ALNS algorithm and the local search procedure under various settings, as described in Section 5.3. The detailed computational results of the ALNS algorithm from solving extensive instances are reported in Table 14, as described in Section 5.5. As shown in Table 3, deviations between the best and average objective values of instances with sizes less than 50 are zero. Note that the ALNS algorithm can optimally solve the instances with 20 requests. Therefore, the corresponding results will not be presented in Table 14.

Table 5
Tuning results of roulette wheel mechanism parameters.

Inst	(π_1, π_2, π_3)						
	(1,5,3)	(1,3,5)	(3,5,1)	(3,1,5)	(5,3,1)	(5,1,3)	(1,1,1)
LRP_20#	653.4	656.6	654.3	655.1	656.6	654.6	655.4
LRP_50#	1403.4	1402.8	1404.1	1404.0	1403.9	1404.6	1402.2
LRP_80#	2255.5	2261.8	2259.7	2256.2	2258.8	2261.6	2258.0
LRP_100#	2858.9	2856.1	2857.0	2860.1	2860.2	2857.0	2855.7
LRP_120#	3287.5	3287.5	3285.6	3292.1	3287.5	3292.0	3297.8
LRP_150#	4218.4	4217.3	4217.3	4218.8	4219.1	4218.8	4218.5
LRP_200#	5309.7	5312.6	5314.4	5307.9	5310.7	5315.7	5307.1
LRP_240#	7219.9	7222.4	7217.0	7219.4	7215.0	7217.2	7215.0
LRP_300#	8770.5	8774.6	8779.2	8774.0	8774.7	8769.3	8773.2
LRP_360#	11493.1	11488.7	11491.3	11492.0	11493.8	11487.8	11492.7
LRP_450#	14286.4	14283.2	14279.6	14282.5	14279.4	14285.5	14287.7
Average	**5614.25**	5614.87	5614.50	5614.74	5614.52	5614.92	5614.85

Table 6
Tuning results of temperature.

Inst	T^{org} =200			T^{org} =150			T^{org} =100		
	Best	*Aver*	*Time*	*Best*	*Aver*	*Time*	*Best*	*Aver*	*Time*
LRP_20#	649	653.4	1.09	649	654.5	1.06	649	653.8	1.13
LRP_50#	1399	1403.4	3.63	1397	1400.5	3.63	1399	1403.7	3.80
LRP_80#	2250	2255.5	8.59	2250	2260.7	8.73	2244	2258.1	8.67
LRP_100#	2851	2858.9	13.18	2847	2855.1	13.18	2847	2859.6	13.45
LRP_120#	3277	3287.5	17.04	3280	3284.5	16.65	3278	3286.0	16.99
LRP_150#	4213	4218.4	32.04	4211	4223.2	32.27	4210	4219.6	32.61
LRP_200#	5300	5309.7	58.96	5300	5314.1	59.13	5307	5313.4	55.98
LRP_240#	7209	7219.9	101.80	7215	7220.8	104.31	7208	7218.6	102.95
LRP_300#	8763	8770.5	174.79	8757	8778.5	172.96	8766	8777.3	180.74
LRP_360#	11477	11493.1	353.52	11478	11496.2	360.13	11473	11482.7	354.61
LRP_450#	14264	14286.4	647.47	14267	14277.5	650.77	14268	14283.1	637.63
Average	5604.73	5614.25	128.37	5604.64	5615.05	129.35	**5604.45**	**5614.17**	128.05

Table 7
Tuning results of cooling rate parameter.

Inst	*cr*						
	0.95	0.97	0.99	0.995	0.997	0.999	0.99975
LRP_20#	653.8	654.3	655.0	652.0	653.0	650.0	649.2
LRP_50#	1403.7	1401.4	1404.3	1403.4	1403.5	1401.0	1401.0
LRP_80#	2258.1	2253.1	2260.0	2257.1	2255.8	2255.8	2254.5
LRP_100#	2859.6	2856.2	2853.3	2856.1	2856.5	2851.9	2852.6
LRP_120#	3286.0	3287.4	3288.0	3292.5	3289.9	3289.6	3290.2
LRP_150#	4219.6	4221.2	4220.9	4218.1	4221.8	4221.5	4222.0
LRP_200#	5313.4	5315.2	5317.4	5325.3	5313.6	5316.5	5321.1
LRP_240#	7218.6	7216.9	7223.2	7215.4	7215.7	7212.6	7216.3
LRP_300#	8777.3	8774.2	8773.5	8775.5	8773.5	8773.6	8778.9
LRP_360#	11482.7	11492.4	11489.0	11494.9	11496.3	11487.2	11494.6
LRP_450#	14283.1	14285.3	14283.1	14276.5	14287.5	14291.9	14289.3
Average	5614.17	5614.33	5615.25	5615.16	5615.19	**5613.78**	5615.43

Table 8
Tuning results of destroy rate parameter.

Inst	μ =0.07		μ =0.1		μ =0.2		μ =0.3		μ =0.4	
	Aver	*Time*	*Aver*	*Time*	*Aver*	*Time*	*Aver*	*Time*	*Aver*	*Time*
LRP_20#	651.0	1.03	650.0	1.07	649.0	1.83	649.0	2.62	649.0	3.05
LRP_50#	1403.9	2.78	1401.0	4.26	1399.6	6.86	1398.8	10.53	1397.4	12.56
LRP_80#	2261.8	5.73	2255.8	9.87	2252.0	16.71	2252.4	78.22	2247.2	37.92
LRP_100#	2856.6	9.88	2851.9	13.52	2847.8	28.41	2848.4	41.39	2848.6	44.91
LRP_120#	3296.7	12.10	3289.6	17.23	3287.0	37.39	3287.2	58.71	3284.8	81.16
LRP_150#	4223.6	21.65	4221.5	33.36	4215.8	73.39	4219.0	115.48	4222.8	172.03
LRP_200#	5324.9	38.00	5316.5	56.43	5309.0	143.43	5307.0	221.43	5316.0	278.45
LRP_240#	7217.3	64.42	7212.6	106.87	7214.2	268.76	7210.6	428.04	7217.0	535.79
LRP_300#	8784.5	112.05	8773.6	172.21	8771.4	445.44	8768.0	645.04	8795.6	844.82
LRP_360#	11498.6	223.42	11487.2	349.71	11500.6	851.36	11524.6	1441.65	11584.4	1880.26
LRP_450#	14293.6	403.48	14291.9	656.32	14319.6	1286.60	14309.0	2181.44	14383.2	2455.65
Average	5619.32	81.32	**5613.78**	129.17	5615.09	287.29	5615.82	474.96	5631.45	576.96

Table 9
Tuning results of the noise parameter.

ϵ	*Best*	*Aver*	*Time*	ϵ	*Best*	*Aver*	*Time*
0.000	5600.82	5613.78	129.17	0.100	5603.00	5610.25	128.05
0.025	5601.00	**5610.36**	128.14	0.150	5603.27	5611.80	128.26
0.050	5602.27	5610.97	128.37	0.200	5603.09	5612.97	128.50
0.070	5602.00	5611.15	129.35	0.300	5604.73	5611.90	128.74

Table 10
Average performance of each removal or insertion operator for all the test instances.

Item	*Insertion operators*								
	—	—	*RSI*	*FFI*	*GI*	*RI_2*	*RI_3*	*RI_4*	*RI_7*
Single	—	—	3.2E-03	2.4E-03	4.5E-03	5.1E-03	5.2E-03	5.3E-03	5.3E-03
Sum	—	—	8.46	2.76	18.31	23.43	22.98	21.40	22.07
Number	—	—	2760.75	1906.21	3822.86	4404.42	4205.69	3938.31	3961.75
Best	—	—	0.96	1.35	2.12	2.47	2.05	1.95	1.77
Better	—	—	8.00	36.31	70.84	88.86	76.96	73.38	72.21
Worse	—	—	879.63	565.85	1767.74	2157.34	1974.26	1839.30	1854.52
Item	*Removal operators*								
	RR	*WR*	*SRR*	*SR*	*TOR*	*DOR*	*EDOR*	*COR*	*HKR*
Single	1.7E-05	1.2E-03	9.2E-06	9.9E-04	2.9E-05	2.4E-05	1.8E-05	9.5E-04	1.5E-05
Sum	0.07	3.79	0.00	2.48	0.11	0.05	0.07	2.94	0.03
Number	4120.92	3596.72	594.73	3304.18	4178.79	1673.05	3553.87	2370.18	1607.55
Best	1.82	1.50	0.35	1.98	1.93	0.88	1.99	1.61	0.63
Better	73.19	64.57	13.92	59.04	69.97	34.53	71.92	62.90	4.95
Worse	1913.70	1548.15	93.37	1230.86	1909.76	531.92	2278.09	1048.76	484.01

Table 11
Average performance of the ALNS algorithm under various settings.

Settings	*Best*	*Aver*	*Time*	*Settings*	*Best*	*Aver*	*Time*
RR	5601.91	5611.55	132.77	*RSI*	5601.73	5611.30	129.18
WR	5602.82	5610.36	132.47	*FFI*	5602.09	5610.85	130.60
SRR	5601.73	5610.43	128.82	*GI*	5601.64	5611.43	127.42
SR	5603.64	5615.57	131.68	*RI_2*	5602.00	5611.58	124.23
TOR	5603.18	5610.88	136.06	*RI_3*	5602.36	5610.79	126.20
DOR	5602.55	5611.15	130.30	*RI_4*	5602.18	5610.55	127.40
EDOR	5602.55	5611.94	127.64	*RI_7*	5602.00	5610.30	126.88
COR	5602.27	5610.94	126.87	*FM*	5601.00	5610.36	128.14
HKR	5601.18	5611.02	129.25	*LNS*	5608.09	5618.75	119.18

Table 12
Average performance of the local search procedure under various settings.

Item	*Initialization*	*FM*	*Disabled neighborhood structure*					
			SRR	*ISRR*	*ISGR*	*IMGR*	*IC*	*IRR*
Time	0.18	0.51	0.48	0.29	0.35	0.31	0.48	0.50
Best	5877.00	5670.18	5686.91	5792.09	5680.45	5774.09	5681.82	5674.55
Aver	5877.00	5683.41	5704.29	5801.96	5700.05	5786.01	5706.96	5697.72

Table 13
Detailed comparison results for different neighbor acceptance criteria.

Inst	*BKS*	LS_I					LS_{II}				
		Best	*Aver*	*Time*	$D_B(\%)$	$D_A(\%)$	*Best*	*Aver*	*Time*	$D_B(\%)$	$D_A(\%)$
LRP_20#	649	649	649.0	1.27	0.00	0.00	649	649.0	1.33	0.00	0.00
LRP_50#	1397	1397	1398.0	4.45	0.00	0.07	1397	1398.4	4.69	0.00	0.10
LRP_80#	2241	2242	2244.9	11.00	0.04	0.17	2241	2243.8	10.56	0.00	0.12
LRP_100#	2841	2841	2844.1	16.87	0.00	0.11	2841	2841.9	15.68	0.00	0.03
LRP_120#	3276	3277	3279.0	21.13	0.03	0.09	3276	3278.6	22.03	0.00	0.08
LRP_150#	4204	4205	4208.1	41.31	0.02	0.10	4204	4208.8	40.24	0.00	0.11
LRP_200#	5287	5291	5295.0	70.02	0.08	0.15	5290	5294.4	69.48	0.06	0.14
LRP_240#	7174	7203	7207.6	135.65	0.40	0.47	7203	7208.0	124.96	0.40	0.47
LRP_300#	8736	8736	8747.7	207.44	0.00	0.13	8739	8750.2	206.00	0.03	0.16
LRP_360#	11446	11449	11455.9	427.47	0.03	0.09	11448	11455.2	423.82	0.02	0.08
LRP_450#	14230	14230	14239.5	738.34	0.00	0.07	14230	14240.1	747.00	0.00	0.07
Average	5589.18	5592.73	5597.16	152.27	0.06	0.13	5592.55	5597.13	151.44	0.05	0.13
Inst	*BKS*	*NONE*					LS_{new}				
		Best	*Aver*	*Time*	$D_B(\%)$	$D_A(\%)$	*Best*	*Aver*	*Time*	$D_B(\%)$	$D_A(\%)$
LRP_20#	649	649	650.4	1.08	0.00	0.22	649	649.0	1.25	0.00	0.00
LRP_50#	1397	1397	1398.5	3.76	0.00	0.11	1397	1397.2	4.01	0.00	0.01
LRP_80#	2241	2244	2252.1	8.55	0.13	0.50	2241	2243.4	9.79	0.00	0.11
LRP_100#	2841	2843	2854.2	13.04	0.07	0.46	2841	2841.8	15.10	0.00	0.03
LRP_120#	3276	3279	3292.6	17.28	0.09	0.51	3276	3277.2	19.57	0.00	0.04
LRP_150#	4204	4212	4218.1	32.48	0.19	0.34	4205	4207.7	36.69	0.02	0.09
LRP_200#	5287	5301	5311.9	58.47	0.26	0.47	5287	5292.8	64.61	0.00	0.11
LRP_240#	7174	7207	7212.6	103.15	0.46	0.54	7174	7200.3	115.53	0.00	0.37
LRP_300#	8736	8756	8773.2	176.23	0.23	0.43	8737	8744.3	187.88	0.01	0.10
LRP_360#	11446	11471	11481.4	351.04	0.22	0.31	11446	11451.1	387.99	0.00	0.04
LRP_450#	14230	14252	14269.0	644.40	0.15	0.27	14231	14239.4	680.63	0.01	0.07
Average	5589.18	5601.00	5610.36	128.14	0.16	0.38	5589.45	5594.93	138.46	0.01	0.09

Table 14
Computational results for all the instances.

Inst	*Best*	*Aver*	*Time*	*D*(%)	*Inst*	*Best*	*Aver*	*Time*	*D*(%)
LRP_50_01	1458	1458.1	4.21	0.01	LRP_80_01	2551	2553.2	9.98	0.09
LRP_50_02	1434	1435.9	3.89	0.13	LRP_80_02	2439	2440.2	10.47	0.05
LRP_50_03	1454	1454.4	4.55	0.03	LRP_80_03	2122	2124.3	9.48	0.11
LRP_50_04	1305	1305.6	4.36	0.05	LRP_80_04	2398	2399.1	9.71	0.05
LRP_50_05	1331	1331.5	4.49	0.04	LRP_80_05	2323	2324.0	10.60	0.04
LRP_50_06	1388	1388.0	4.53	0.00	LRP_80_06	2090	2090.3	10.03	0.01
LRP_50_07	1285	1285.2	3.90	0.02	LRP_80_07	2177	2178.6	9.32	0.07
LRP_50_08	1302	1306.5	3.81	0.35	LRP_80_08	2287	2288.6	10.09	0.07
LRP_50_09	1465	1465.0	4.03	0.00	LRP_80_09	2328	2329.3	9.91	0.06
LRP_50_10	1303	1303.7	4.32	0.05	LRP_80_10	2458	2458.1	9.47	0.00
LRP_100_01	2829	2831.6	16.44	0.09	LRP_120_01	3446	3448.3	23.69	0.07
LRP_100_02	2884	2884.8	15.36	0.03	LRP_120_02	3632	3633.0	22.47	0.03
LRP_100_03	2973	2973.2	15.81	0.01	LRP_120_03	3673	3673.6	22.71	0.02
LRP_100_04	2838	2838.0	15.48	0.00	LRP_120_04	3417	3421.0	19.76	0.12
LRP_100_05	2796	2800.6	13.61	0.16	LRP_120_05	3495	3498.2	23.04	0.09
LRP_100_06	2973	2974.8	15.92	0.06	LRP_120_06	3507	3509.0	22.96	0.06
LRP_100_07	2665	2666.7	13.60	0.06	LRP_120_07	3587	3589.0	21.23	0.06
LRP_100_08	2977	2978.8	15.56	0.06	LRP_120_08	3787	3789.2	22.57	0.06
LRP_100_09	2704	2710.1	13.49	**0.23**	LRP_120_09	4066	4070.1	23.63	0.10
LRP_100_10	3081	3083.8	15.65	0.09	LRP_120_10	3315	3317.7	20.44	0.08
LRP_150_01	4319	4322.0	36.01	0.07	LRP_200_01	5694	5700.6	71.09	0.12
LRP_150_02	4322	4323.7	35.97	0.04	LRP_200_02	5785	5792.4	69.82	0.13
LRP_150_03	4614	4619.0	38.09	0.11	LRP_200_03	5321	5326.8	70.56	0.11
LRP_150_04	4371	4375.5	39.17	0.10	LRP_200_04	5550	5554.5	74.99	0.08
LRP_150_05	4560	4561.4	37.90	0.03	LRP_200_05	5720	5722.4	67.78	0.04
LRP_150_06	4299	4303.5	38.49	0.10	LRP_200_06	5667	5671.8	76.26	0.08
LRP_150_07	4556	4560.8	39.49	0.11	LRP_200_07	5787	5788.1	67.97	0.02
LRP_150_08	4628	4631.3	34.82	0.07	LRP_200_08	5465	5468.7	73.12	0.07
LRP_150_09	4631	4634.8	38.29	0.08	LRP_200_09	5990	5992.9	74.36	0.05
LRP_150_10	4407	4409.6	36.48	0.06	LRP_200_10	5543	5545.3	74.68	0.04
LRP_240_01	6944	6948.1	119.40	0.06	LRP_300_01	9092	9097.5	215.09	0.06
LRP_240_02	7062	7066.5	122.92	0.06	LRP_300_02	9365	9371.7	201.85	0.07
LRP_240_03	7045	7050.6	118.98	0.08	LRP_300_03	9323	9329.8	223.52	0.07
LRP_240_04	7101	7109.4	114.08	0.12	LRP_300_04	8543	8547.7	182.16	0.06
LRP_240_05	7096	7100.3	121.05	0.06	LRP_300_05	8877	8881.4	219.82	0.05
LRP_240_06	6715	6718.8	117.22	0.06	LRP_300_06	8995	9000.8	212.74	0.06
LRP_240_07	7288	7290.9	121.87	0.04	LRP_300_07	9127	9131.3	226.56	0.05
LRP_240_08	6963	6968.3	114.45	0.08	LRP_300_08	9327	9331.0	197.52	0.04
LRP_240_09	6915	6919.0	115.01	0.06	LRP_300_09	9302	9308.6	231.26	0.07
LRP_240_10	6854	6857.3	120.92	0.05	LRP_300_10	8830	8836.8	213.69	0.08
LRP_360_01	10411	10416.3	364.22	0.05	LRP_450_01	13560	13566.0	731.94	0.04
LRP_360_02	10041	10049.5	388.97	0.08	LRP_450_02	13792	13798.9	640.24	0.05
LRP_360_03	11252	11256.9	374.66	0.04	LRP_450_03	13508	13512.3	750.52	0.03
LRP_360_04	10766	10770.6	404.36	0.04	LRP_450_04	14255	14263.3	738.84	0.06
LRP_360_05	10290	10298.5	392.71	0.08	LRP_450_05	13938	13948.9	784.85	0.08
LRP_360_06	10784	10789.2	398.49	0.05	LRP_450_06	13274	13280.9	694.45	0.05
LRP_360_07	10149	10157.3	334.55	0.08	LRP_450_07	14114	14129.7	673.84	0.11
LRP_360_08	10563	10574.7	332.54	0.11	LRP_450_08	13406	13412.4	748.43	0.05
LRP_360_09	10965	10976.1	331.85	0.10	LRP_450_09	13777	13791.0	672.94	0.10
LRP_360_10	11854	11876.2	329.54	0.19	LRP_450_10	14126	14135.1	747.34	0.06
Average	5589.84	5593.66	143.36	0.06					